Motion Drive

Variateur numérique pour moteur Brushless
Série MD

Manuel d'utilisation

Lire attentivement ce manuel avant la mise en route et respecter toutes les indications avec le symbole :

SERAD SAS
271, route des crêtes
44440 TEILLE – France
☎ +33 (0)2 40 97 24 54
+33 (0)2 40 97 27 04
http://www.serad.fr
info@serad.fr

SOMMAIRE

1- Introduction

1-1- Mise en garde

Le montage, le raccordement, la mise en service et la maintenance de l'appareil ne peuvent être réalisés que par des personnes qualifiées.

Il est indispensable de respecter les instructions de sécurité. Des blessures et dommages corporels peuvent résulter d'une méconnaissance de ces instructions de sécurité.

Une mauvaise mise à la terre du variateur peut endommager ses composants électroniques.

Les règles de prévention des accidents sont les suivantes :

- VDE 0100 — Spécification pour l'installation des systèmes de puissance jusqu'à 1000 V
- VDE 0113 — Equipement électrique de machines
- VDE 0160 — Equipement de système de puissance avec des composants électroniques

- ***Ne jamais ouvrir l'appareil.***
- ***Des hautes tensions pouvant être dangereuses sont appliquées à l'intérieur du variateur et des connecteurs. Pour cela, débrancher le variateur et attendre au moins 5 minutes pour que les condensateurs se déchargent avant de débrancher un connecteur.***
- ***Ne jamais débrancher ou brancher de connecteurs sous tension.***
- ***L'appareil peut comporter des surfaces très chaudes.***
- ***Les variateurs MD230 ne doivent pas être alimentés en régime neutre IT, au risque de détérioration de l'appareil.***

Ne pas manipuler l'appareil de façon inappropriée sous peine de détérioration de certains composants électroniques par décharges électrostatiques.

1-2- Description du variateur MD

1-2-1- Généralités :

Les variateurs intelligents brushless série MD sont tout spécialement adaptés aux dynamiques élevées.

Ils intègrent l'alimentation, la résistance de freinage et le filtre réseau.

Ils peuvent être utilisés en mode couple, en mode vitesse, en mode positionnement.

Les bus de communications MODBUS et CANopen assurent des configurations en réseau.

Grâce à leur langage Basic multitâches, leurs fonctions de MOTION et automate intégrées, ils répondent aux applications les plus diverses.

1-2-2- Données techniques :

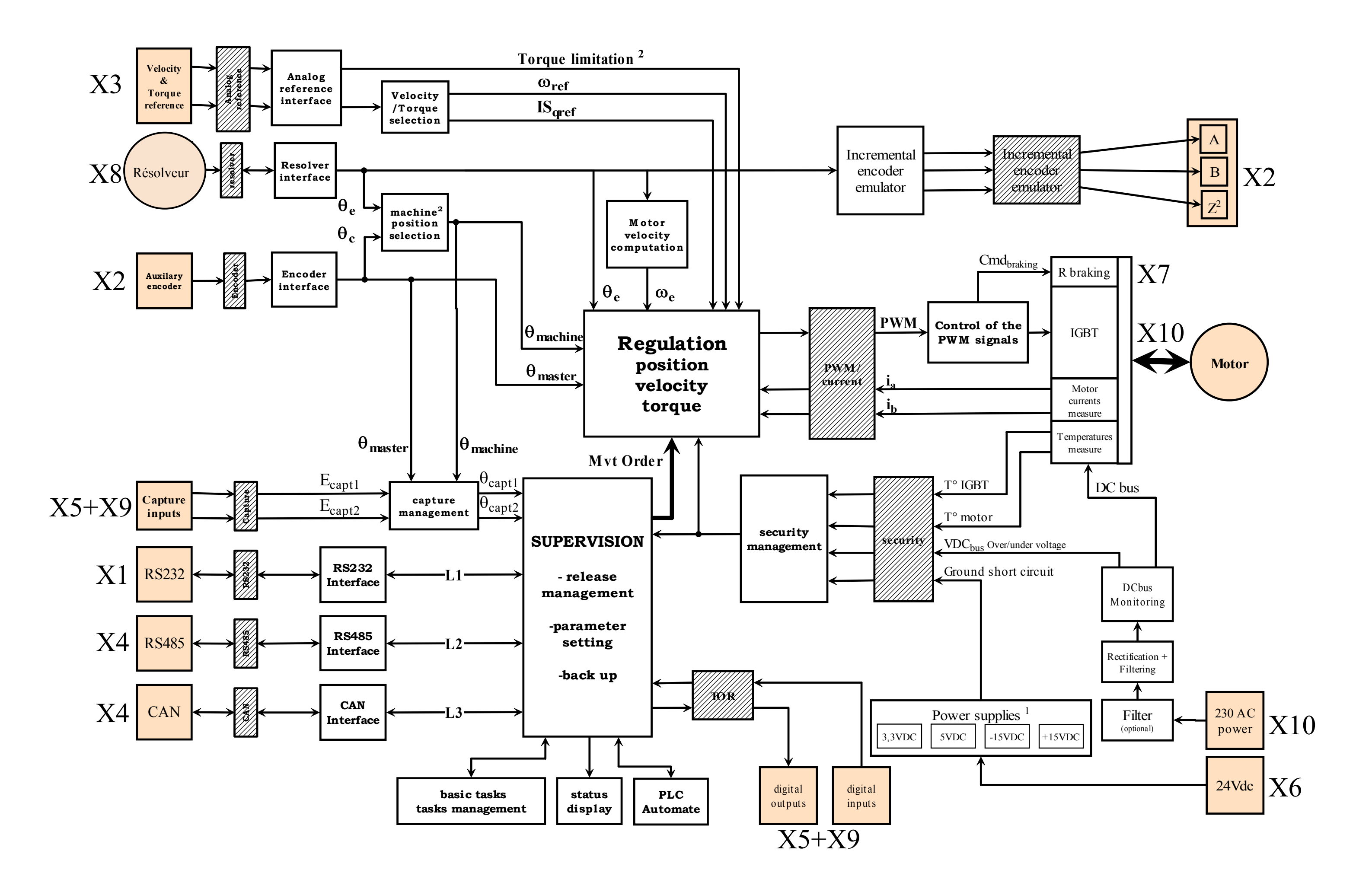
<table>
<tr><td>Alimentation :</td><td>MD 230 M : 230V AC ±10% monophasée
MD 400 T : 400V AC ±10% triphasée</td></tr>
<tr><td>Alimentation auxiliaire :</td><td>24 V DC ±10% 0,5A typique 0,7A maxi si retour codeur</td></tr>
<tr><td>Filtre réseau :</td><td>Intégré</td></tr>
<tr><td>Fréquence de découpage :</td><td>6.25 KHz, commande sinusoïdale du moteur</td></tr>
<tr><td>Tension DC Bus :</td><td>310 V pour série MD 230, 560V pour série MD 400</td></tr>
<tr><td>Courant de fuite :</td><td>2,2 mA si MD 230, 1 mA si MD 400</td></tr>
<tr><td>Résistance de freinage :</td><td>Intégrée : MD 230 : 110 ohms 30W
MD 400 : 180 ohms 30W
Possibilité d'ajouter une résistance externe :
<table>
<tr><td>Type</td><td>Valeur Min.</td><td>Puissance Cont. Max.</td><td>Puissance Imp. Max</td></tr>
<tr><td>MD230/1 ou /2</td><td>60 Ω</td><td>1000W</td><td>2300W</td></tr>
<tr><td>MD230/5 ou /7</td><td>30 Ω</td><td>1800W</td><td>4600W</td></tr>
<tr><td>MD 400</td><td>80 Ω</td><td>2800W</td><td>7000W</td></tr>
</table></td></tr>
<tr><td>Protections :</td><td>Court-circuit entre phases, phase à la terre, surcourant, I^2t
Surtension, sous-tension
Défaut feedback moteur</td></tr>
</table>

Retour moteur :	Résolveur (résolution 16 bits) Précision absolue résolveur ± 0,7° Codeur incrémental (option)
Codeur maître auxiliaire :	Incrémental : A, /A, B, /B, Z, /Z Fréquence maxi : 800 KHz
Emulation codeur :	Incrémental : A, /A, B, /B, Z, /Z 1024 points par tour
Diagnostic :	Afficheur 7 segments
Communication :	RS 232 MODBUS RTU RS 422 (point à point), RS 485 MODBUS RTU (option) CANopen (option)
Entrées logiques :	4 voies en standard, 12 voies sur module d'extension : type : PNP 24 Vdc, 12mA par voie niveau logique 0 : de 0 à 5 V niveau logique 1 : de 10 à 30 V
Sorties logiques :	2 voies en standard : S1 : relais, 48 Vdc maxi, 48 Vac maxi, 3 A maxi S2 : statique NPN (collecteur ouvert) 24 Vdc, 100 mA 8 voies sur module d'extension : type : statique PNP 24 Vdc, 100 mA maxi par voie protection contre les court-circuits et surchauffe
Entrées analogiques :	2 voies : Tension d'entrée : ±10 V Tension d'entrée maxi: ±12 V Impédance d'entrée : 20 Kohm Résolution : 10 bits
Sortie analogique :	1 voie : Tension de sortie : ±10 V Courant de sortie maxi: 5 mA Résolution : 8 bits

Architecture :	Processeur DSP 40 MHz Mémoire FLASH pour stockage des programmes et paramètres Mémoire RAM pour stockage des données Noyau temps réel multitâches
Boucles de régulation :	Boucle de courant : 160 µs Boucle de vitesse : 320 µs Boucle de position : 640µs
Modes de fonctionnement :	Mode couple Mode vitesse Mode positionnement Fonctions MOTION
Température de service :	0 à 40°C
Température de stockage :	-10 à 70°C
Indice de protection :	IP 20

Drive	Courant nominal	Courant crête (2s)	Puissance nominale	Dimensions l x h x p
MD 230 / 1	1,25 Aeff	2,5 Aeff	0,35 kVA	67 x 215 x 203
MD 230 / 2	2,5 Aeff	5 Aeff	0,7 kVA	67 x 215 x 203
MD 230 / 5	5 Aeff	10 Aeff	1,5 kVA	67 x 215 x 203
MD 230 / 7	7,5 Aeff	15 Aeff	2,3 kVA	67 x 215 x 203
MD 400 / 1	1,25 Aeff	2,5 Aeff	0,7 kVA	67 x 215 x 203
MD 400 / 2	2,5 Aeff	5 Aeff	1,4 kVA	67 x 215 x 203
MD 400 / 5	5 Aeff	10 Aeff	3 kVA	67 x 215 x 203

Schéma synoptique :



1-3- Description du logiciel DPL

1-3-1- Généralités :

L'atelier logiciel DPL, grâce à son outil graphique, permet de configurer très facilement le variateur à partir d'un PC.

Sous environnement Windows, il offre une convivialité parfaite, des écrans avec multi-fenêtrage et une aide complète.

Les fonctions d'auto tuning, générateur de trajectoires et oscilloscope assurent une mise en œuvre rapide et optimale.

1-3-2- Données techniques :

↳ Configuration de tous les paramètres par groupe : moteur, régulation, codeur, E/S analogiques, E/S logiques, communication, sécurités…

↳ Affichage des paramètres d'états : vitesses, courants, couples, positions…

↳ Sauvegarde et impression des paramètres sur PC

↳ Fonctions d'auto-tuning résolveur

↳ Générateur de trajectoires : position, accélération, décélération, vitesse

↳ Oscilloscope numérique multi-canaux

↳ Tableau de bord : axe, entrées / sorties

↳ Reconnaissance automatique du variateur connecté

↳ Travail possible en mode non connecté (vérification et édition de paramètres…)

↳ Aide en ligne pour chaque fenêtre

1-3-3- Langage de programmation DPL :

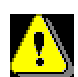
Les variateurs de la série MD intègrent un noyau temps réel multitâches et plus de 1000 variables utilisateurs.

Le langage motion-basic DPL permet à l'utilisateur de développer, tester et sauvegarder ses propres programmes applicatifs.

Les applications peuvent être toute sorte de combinaison de mode couple, mode vitesse et mode positionnement. Les entrées / sorties sont librement utilisées dans le programme, ainsi que les paramètres et les variables.

2- Installation

2-1- Généralités

 **Il est très important de respecter les points suivants :**

🔖 Une mauvaise mise à la terre du variateur peut endommager ses composants électroniques.

🔖 Le variateur doit être installé verticalement pour assurer un refroidissement naturel par convection.

🔖 Il doit être à l'abri de l'humidité, des projections de liquides quelconques, de la poussière.

🔖 Les câbles résolveur, moteur, codeur devront être blindés, la tresse étant reliée de chaque côté au châssis.

🔖 Le câble consigne analogique devra être blindé, la tresse étant reliée de chaque côté au châssis.

🔖 Le câble de liaison série RS 232 variateur / PC devra être blindé, la tresse étant reliée de chaque côté au châssis. Il devra être débranché du variateur lorsqu'il n'est plus utilisé. Tous ces câbles, ainsi que les câbles d'entrées-sorties, devront être séparés et éloignés des circuits de puissance.

🔖 Il faut prévoir sur toutes les sorties statiques (Q2 à Q10) des diodes de roue libre sur les charges inductives. Ces diodes doivent être placées le plus près possible de la charge. Les conducteurs d'alimentation et de signaux ne doivent pas être le siège de surtensions.

🔖 Les normes de sécurité imposent un réarmement manuel après un arrêt provoqué soit par coupure secteur, par appui sur l'arrêt d'urgence, par défaut sortie « drive ready ».

🔖 Sur tout défaut grave, il est obligatoire de couper l'alimentation de puissance du variateur.

🔖 La sortie « drive ready » devra être reliée en série dans la boucle d'arrêt d'urgence.

🔖 Dans le cas d'un axe fini, les capteurs de limitation de la course devront être reliés sur les entrées fin de course ou en série dans la boucle d'arrêt d'urgence

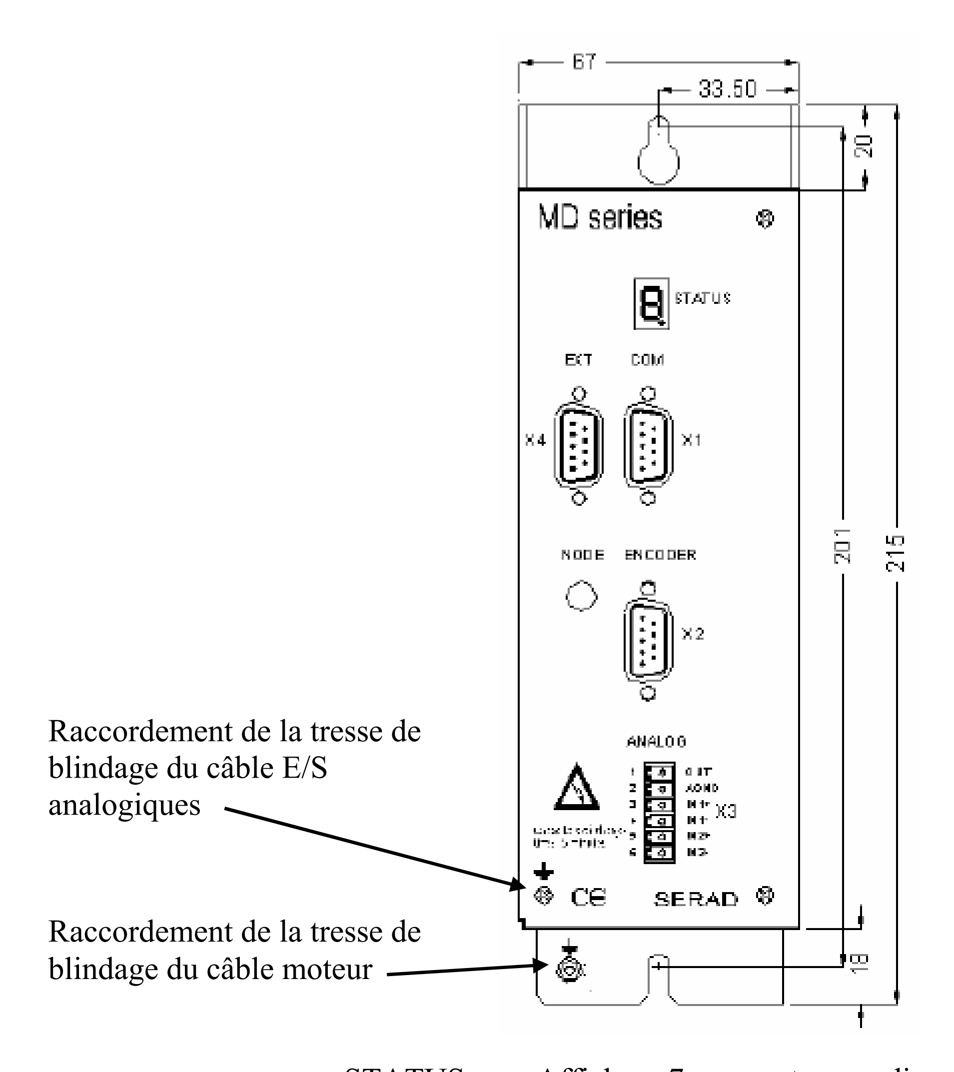
🔖 Si le variateur est configuré en mode couple ou vitesse, la validation du variateur faite à partir de l'entrée ENABLE devra être gérée par l'appareil en amont (commande d'axes, automate …)

🔖 Si le variateur est configuré en mode position, le paramètre "Erreur de poursuite maxi" devra être réglé.

🔖 Si le variateur contient un programme applicatif développé à partir du langage DPL, relier l'information « Puissance armoire électrique OK » sur une entrée

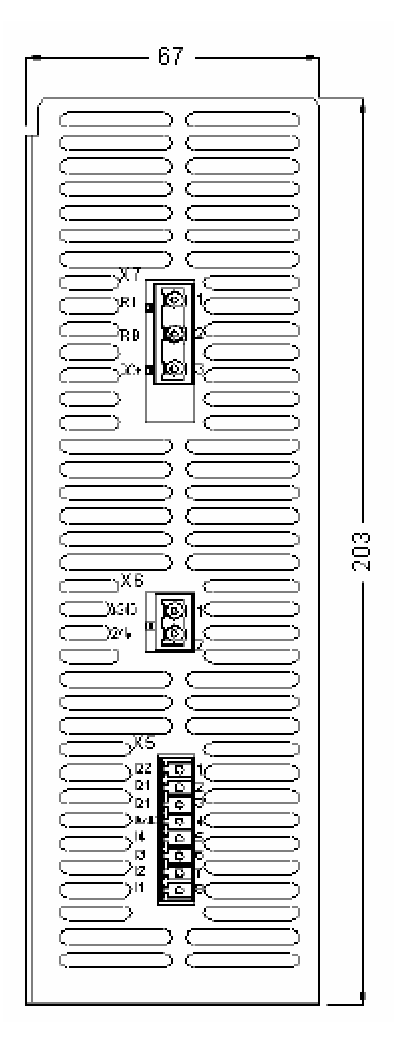
automate et la traiter dans une tâche basic non bloquante de sécurité. Sur détection d'une erreur de poursuite, le variateur passe en boucle ouverte et ouvre la sortie « drive ready ».

2-2- Vue de face



	STATUS	Afficheur 7 segments pour diagnostic
X1	COM	Port de communication RS 232 pour paramétrage PC
X2	ENCODER	Entrée codeur maître / Sortie émulation codeur
X3	ANALOG	Entrées et sorties analogiques
X4	EXT	Extension: Bus de communication optionnel

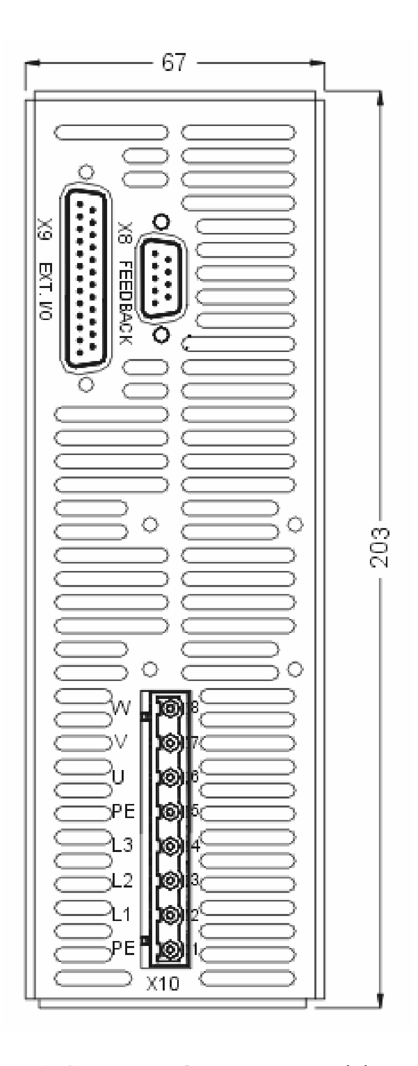
2-3- Vue de dessus



X5	I/O	Entrées et sorties logiques
X6	24Vdc	Alimentation auxiliaire 24 Vdc
X7	RB	Résistance de freinage externe

⚠ La tension sur le connecteur X7 peut atteindre 400V pour un MD 230 et 800V pour un MD 400!

2-4- Vue de dessous

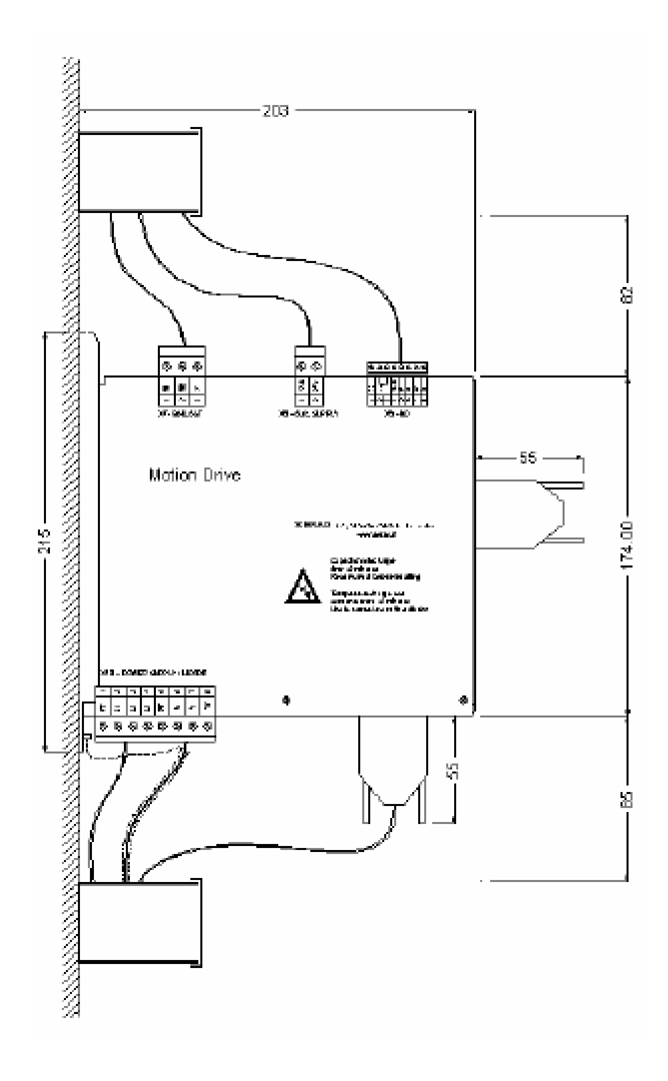


X8	FEEDBACK	Entrée retour position moteur (résolveur, codeur)
X9	EXT I/O	Option : Extension d’entrées / sorties logiques
X10	POWER	Alimentation monophasée ou triphasée Alimentation 3 phases moteur

Attention au câblage du connecteur X10. Une mauvaise connexion peut endommager gravement le variateur. X10 comporte également des tensions dangereuses.

2-5- Montage

On peut monter de nombreux variateurs les uns à côté des autres en respectant les espaces de séparation pour une bonne convection naturelle (laisser un espace minimum de 20 mm entre deux variateurs), la mise en place des connecteurs et le passage des câbles.

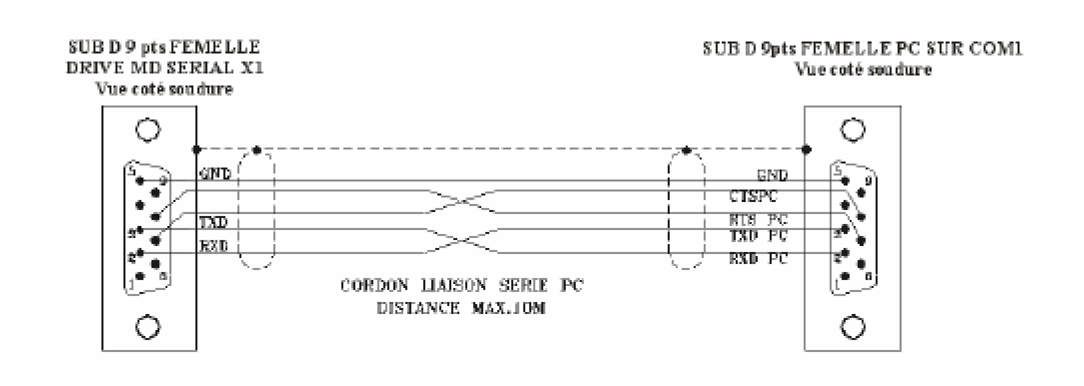


2-6- Affectation et brochages des connecteurs

X1: Port de communication RS 232 pour paramétrage PC

Connecteur SUBD 9 points mâle

N°	Nom	Type	Description
1			
2	RXD	Inp	Réception des données
3	TXD	Out	Transmission des données
4			
5	GND		0V
6			
7			
8	CTS	Inp	Activation liaison système
9			
	SHIELD		Raccordement de la tresse blindée sur le corps du SUBD



X2: Entrée codeur maître / sortie émulation codeur

Connecteur SUBD 9 points femelle

N°	Nom	Type	Description
1	A	I/O	Voie A
2	/A	I/O	Voie A complémentée
3	B	I/O	Voie B
4	/B	I/O	Voie B complémentée
5	Z	I/O	Voie Z
6	/Z	I/O	Voie Z complémentée
7	+5Vdc	Out	Alimentation pour codeur externe 100 mA maxi
8	GND		0V
9			
⚠	**SHIELD**		**Raccordement de la tresse blindée sur le corps du SUBD**

X3: Entrées / sorties analogiques

Codeur 5V TTL (0-5V, différentiel)

Connecteur débrochable 6 points au pas de 3,81 mm

N°	Nom	Type	Description
1	OUT	Out	Sortie analogique fonction monitoring
2	AGND		0V analogique
3	IN1+	Inp	Entrée analogique 1 : consigne vitesse ou couple suivant le mode
4	IN1-	Inp	Entrée analogique 1
5	IN2+	Inp	Entrée analogique 2 : consigne limitation de couple
6	IN2-	Inp	Entrée analogique 2
⚠	**SHIELD**		**Raccordement de la tresse blindée sur la vis prévue en face avant du boîtier du variateur**

X4: Extension: Bus de communication optionnel

N°	Module RS 232 SUBD 9 pts mâle	Module RS 422 SUBD 9 pts femelle	Module RS 485 SUBD 9 pts femelle	Module CANopen SUBD 9 pts femelle
1				
2	RXD			
3	TXD	RX-		
4		RX+		
5	GND	GND	GND	GND
6				
7		TX-	TRX-	CAN_L
8		TX+	TRX+	CAN_H
9				
⚠	**SHIELD - Raccorder la tresse blindée sur le corps du SUBD**			

- Numéro d'adresse : Pour les modules RS422, RS485 et CANopen, le NodeID correspond à la position de la roue codeuse + 1

  Ex : roue codeuse en position 3 ⇒ NodeID 4

  Numéro d'adresse étendue : relier la pin 1 à la pin 6. Le NodeID correspond alors à la position de la roue codeuse + 17

  Ex : roue codeuse en position 3 ⇒ NodeID 20

- Validation des résistances de terminaison du bus (120Ω) :

  Pour le module RS422, relier la pin 2 à la pin 3, relier la pin 8 à la pin 9.

  Pour le module RS485 et CANopen, relier la pin 8 à la pin 9.

X5: Entrées / sorties logiques

Connecteur débrochable 8 points au pas de 3,81 mm

N°	Nom	Type	Description
1	Q2	Out	Sortie 2 programmable : **type NPN *** statique 24 Vdc 100mA
2	Q1	Out	Sortie 1 programmable : fonction DRIVE READY en standard
3	Q1		Type relais contact NO entre les bornes 2 et 3

4	DGND		0V entrées / sorties logiques
5	I4	Inp	Entrée 4 programmable
6	I3	Inp	Entrée 3 programmable
7	I2	Inp	Entrée 2 programmable
8	I1	Inp	Entrée 1 programmable: fonction ENABLE en standard

La sortie Q2* type collecteur ouvert : retour de 0V ⇒ la charge doit être branchée entre Q2 et le +.

X6: Alimentation auxiliaire 24 Vdc

Connecteur débrochable 2 points au pas de 5,08 mm

N°	Nom	Type	Description
1	XGND		0V
2	24Vdc	Inp	Alimentation carte, backup position moteur

X7: Résistance de freinage externe

Connecteur débrochable 3 points au pas de 7,62 mm

N°	Nom	Type	Description
1	RI		Résistance de freinage interne *
2	RB		Résistance de freinage *
3	DC Bus +	Out	Bus continu (310 V sur MD 230, 560 V sur MD 400)

*Sélection de la résistance de freinage :

- Résistance interne : Mettre un shunt entre les bornes 1 et 2

- Résistance externe : Enlever le shunt entre les bornes 1 et 2

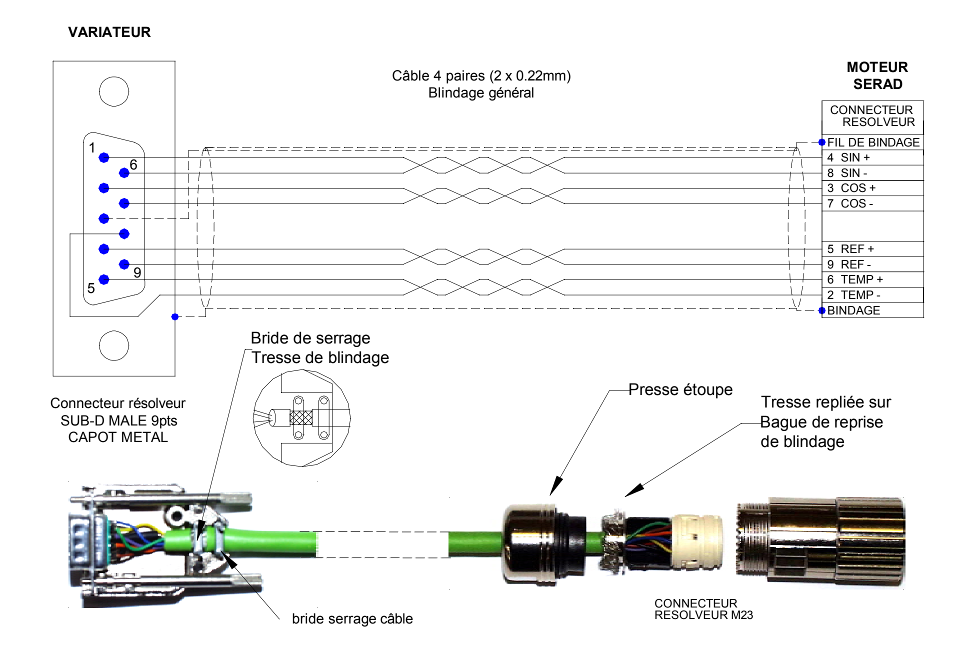
Raccorder la résistance externe entre les bornes 2 et 3

La tension sur le connecteur X7 peut atteindre 400V pour un MD 230 et 800V pour un MD 400!

X8: Entrée retour position moteur (résolveur)

Connecteur SUBD 9 points femelle

N°	Nom	Type	Description
1	S2	Inp	Voie sinus
2	S1	Inp	Voie cosinus
3	AGND		0V analogique
4	R1	Out	Excitation
5	°CM+	Inp	Capteur température moteur
6	S4	Inp	Référence voie sinus
7	S3	Inp	Référence voie cosinus
8	°CM-	Inp	Référence capteur température moteur
9	R2	Out	Référence excitation
⚠	**SHIELD**		**Raccordement de la tresse blindée sur le corps du SUBD**



⚠ **La longueur maximum des câbles résolveur et moteur est de 20m, au-delà de cette longueur, veuillez prendre contacte avec notre support technique.**

X9: Option : Extension 12 entrées / 8 sorties logiques

Connecteur SUBD 25 points femelle

N°	Nom	Type	Description
1	I5	Inp	Entrée 5 programmable
2	I6	Inp	Entrée 6 programmable
3	I7	Inp	Entrée 7 programmable
4	I8	Inp	Entrée 8 programmable
5	I9	Inp	Entrée 9 programmable
6	I10	Inp	Entrée 10 programmable
7	IOGND*		0V entrées / sorties logiques
8	Q3	Out	Sortie 3 programmable
9	Q4	Out	Sortie 4 programmable
10	Q5	Out	Sortie 5 programmable
11	Q6	Out	Sortie 6 programmable
12	IO 24Vdc**	Inp	Alimentation externe 24 Vdc
13	IO 24Vdc**	Inp	Alimentation externe 24 Vdc
14	I11	Inp	Entrée 11 programmable
15	I12	Inp	Entrée 12 programmable
16	I13	Inp	Entrée 13 programmable
17	I14	Inp	Entrée 14 programmable
18	I15	Inp	Entrée 15 programmable
19	I16	Inp	Entrée 16 programmable
20	Q7	Out	Sortie 7 programmable
21	Q8	Out	Sortie 8 programmable
22	Q9	Out	Sortie 9 programmable
23	Q10	Out	Sortie 10 programmable
24	IOGND*		0V entrées / sorties logiques
25	IOGND*		0V entrées / sorties logiques
	SHIELD		**Raccordement de la tresse blindée sur le corps du SUBD**

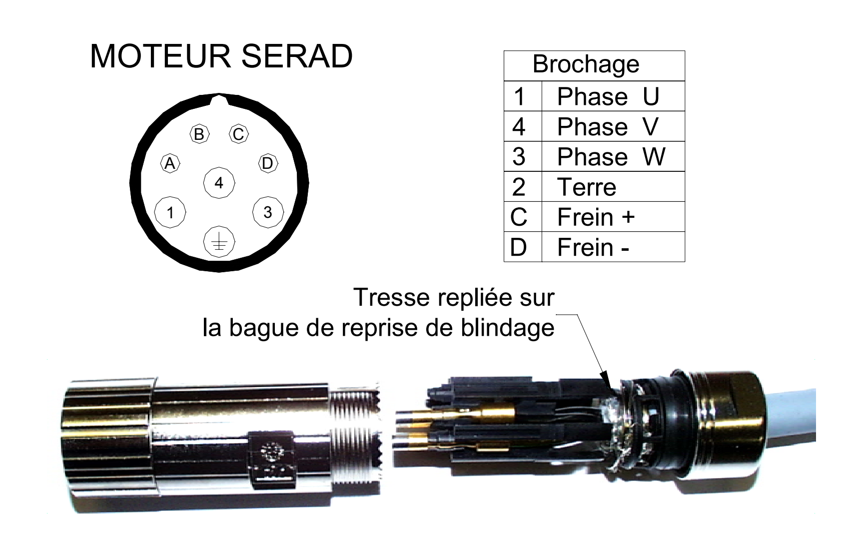
*Pins 7, 24 et 25 : connexion interne **Pins 12, 13 : connexion interne

X10: Alimentation réseau, alimentation moteur

Connecteur débrochable 8 points au pas de 7,62 mm

N°	Nom	Type	Description
1	PE		Terre réseau
2	L1*	Inp	Phase L1 réseau 230V pour MD 230, 400V pour MD 400
3	L2*	Inp	Phase L2 réseau 230V pour MD 230, 400V pour MD 400
4	L3	Inp	Phase L3 réseau 230V pour MD 230, 400V pour MD 400
5	PE		Terre moteur
6	U	Out	Phase U moteur
7	V	Out	Phase V moteur
8	W	Out	Phase W moteur

Pour un réseau 230 Vac monophasé, raccorder la phase sur L1 et le neutre sur L2



⚠ Attention au câblage du connecteur X10. Une mauvaise connexion peut endommager gravement le variateur. X10 comporte également des tensions dangereuses.

Le câble blindé moteur doit arriver directement sur les bornes du variateur.

Relier la tresse de blindage sur la vis prévue à cet effet (voir 2-2 Vue de face).

La longueur maximum des câbles résolveur et moteur est de 20m, au-delà de cette longueur, veuillez prendre contacte avec notre support technique.

2-7- Câbles

Nous vous proposons tous les câbles avec connecteurs montés. Ceux-ci sont disponibles en différentes qualités (standard, compatible chaîne porte câble, etc.), nous consulter.

- Câble COM de communication RS 232 X1 :

  Câble blindé, 4 fils

  Tresse de blindage reliée à chaque extrémité au capot des SUBD.

- Câble ENCODER X2 :

  Câble avec blindage général, 4 paires torsadées 0.25 mm²

  Tresse de blindage reliée à chaque extrémité au capot des SUBD.

- Câble ANALOG X3 :

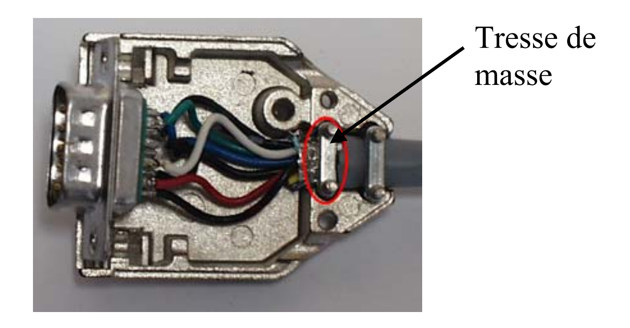
  Câble blindé 2 fils 0.25 mm² par entrée analogique.

  Tresse de blindage à relier coté variateur sur la vis prévue à cet effet (voir 2-2 Vue de face) et l'autre côté au châssis de l'appareil (exemple : commande d'axes …).

- Câble FEEDBACK retour moteur (resolver) X8 :

  Câble avec blindage général, 4 paires torsadées 0.25 mm²

  Raccordement de la tresse de masse au SUBD résolveur comme sur la photo ci-dessous :



Tresse de masse

- Câble POWER moteur X10 :

  Câble avec blindage général 4 fils (plus deux si frein).

  Section 1,5 mm² pour variateur jusqu'à 8A. Au delà, prévoir du 2,5 mm².

  Tresse de blindage à relier côté variateur sur la vis prévue à cet effet (voir 2-2 Vue de face).

⚠ **La longueur maximum des câbles résolveur et moteur est de 20m, au-delà de cette longueur, veuillez prendre contacte avec notre support technique.**

2-8- Schémas de raccordement

Toutes les connexions doivent être réalisées par des personnes qualifiées. Les câbles doivent être testés avant d'être connectés, toute mauvaise connexion peut entraîner de graves dysfonctionnements.

Mettre hors tension le variateur avant d'insérer ou de retirer des connecteurs.

S'assurer que la borne de terre du connecteur de l'alimentation du variateur est bien connectée (borne 1 du connecteur X10).

Connecter la terre du moteur au point de terre du variateur (borne 5 du connecteur X10) avant toute mise sous tension.

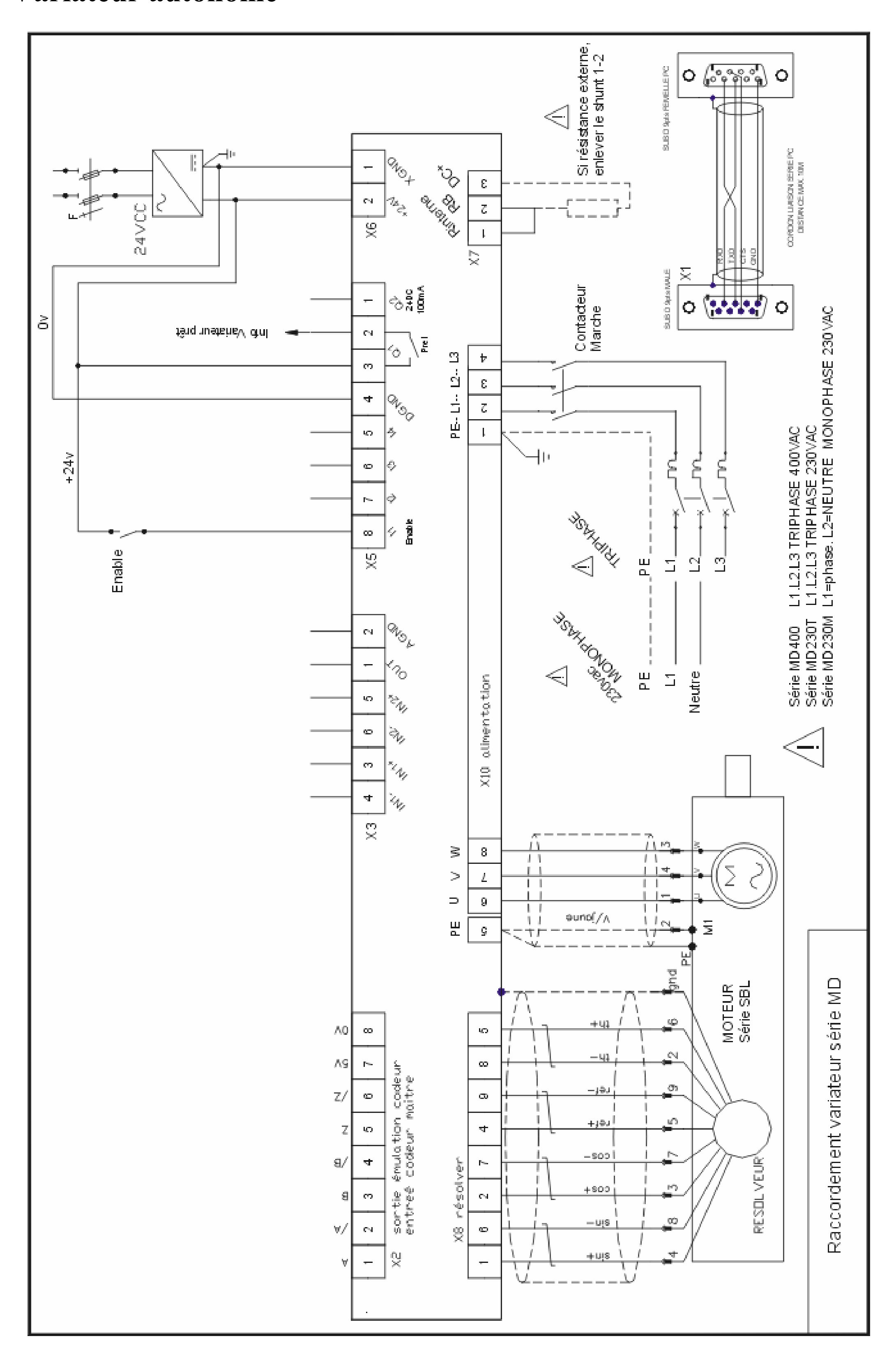
Pour les câbles blindés, raccorder la tresse au châssis à chaque extrémité via les capots des connecteurs 'pour les SUBD) ou les vis prévues à cet effet (connecteurs X3, X10) afin d'assurer une équipotentialité optimale.

Toute bobine (frein) alimentée par courant continu (24V) doit être obligatoirement pourvue d'une diode de roue libre (ex : 1N4007) afin d'empêcher des surtensions (plus de 80V) qui risqueraient de détériorer l'ensemble de l'électronique.

Drive	Tension d'entrée	Courant d'entrée max	Protection : Disjoncteur courbe C	Section câble
MD230/1	230V monophasé	3,5A	10A maxi	1,5²
MD230/2	230V monophasé	7A	10A maxi	1,5²
MD230/5	230V monophasé	14A	10A maxi	1,5²
MD230/7	230V monophasé	21A	16A maxi	2,5²
MD400/1	400V triphasé	2,2A	10A maxi	1,5²
MD400/2	400V triphasé	4,2A	10A maxi	1,5²
MD400/5	400V triphasé	8,2A	10A maxi	1,5²

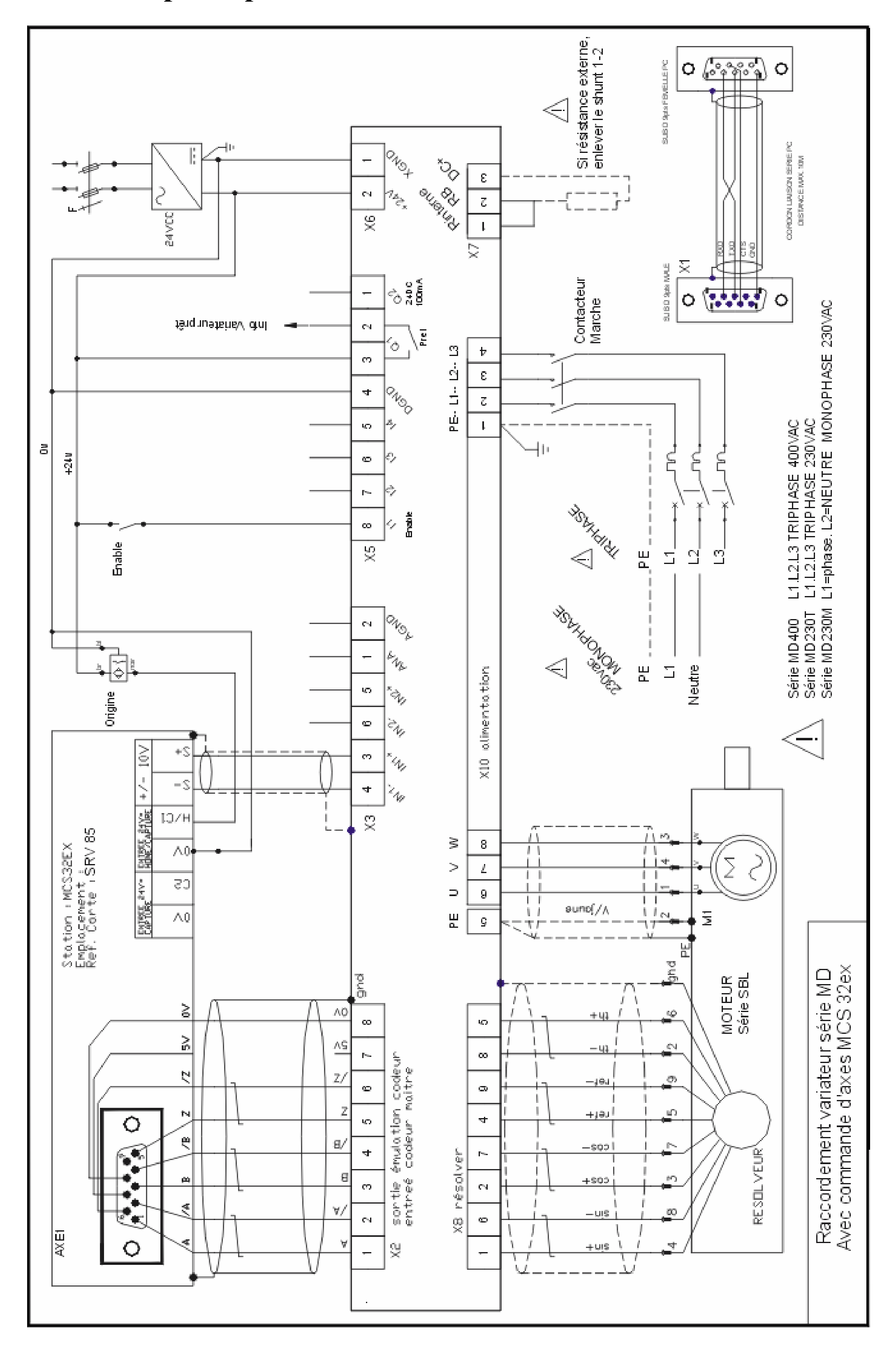
Attention : le courant d'appel pour chaque variateur est de 25A

2-9- Variateur autonome



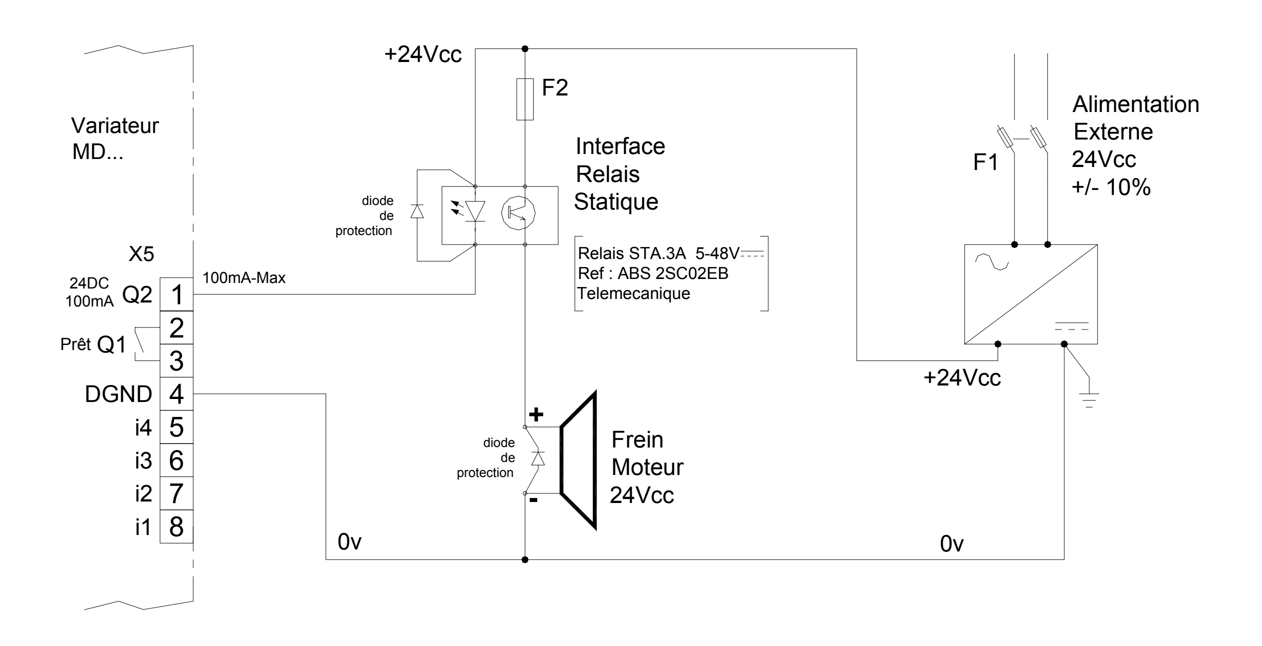
La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +24Vdc.

2-10- Variateur piloté par une commande d'axe



La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +24Vdc.

2-11- Raccordement d'un frein moteur



La sortie Q2 est du type NPN (collecteur ouvert) 100 mA maxi. La charge doit être branchée entre Q2 et le +24Vdc.

A partir du logiciel DPL de paramétrage, aller dans le menu Paramètres / Entrées-Sorties digitales et sélectionner la fonction Frein dans la sortie n°2

Il est obligatoire de mettre les 2 diodes de protection sous peine d'endommager les composants internes du variateur.

2-12- Vérifications avant mise en route

↳ L'entrée ENABLE étant à 0, mettre sous tension l'alimentation auxiliaire 24 Vdc.

↳ S'assurer que l'afficheur de STATUS s'allume.

↳ Mettre la puissance.

↳ Si l'afficheur de STATUS indique un message d'erreur (se reporter à la liste des erreurs).

3- Logiciel DPL

3-1- Installation du logiciel DPL

3-1-1- Configuration du système

- **Configuration minimale :**

  ⇒ PC Pentium

  ⇒ RAM 32 Mo

  ⇒ Disque dur (35 Mo disponibles)

  ⇒ Microsoft® Windows™ 95, 98 , NT, 2000 et XP

  ⇒ Lecteur de CD-ROM (2X)

  ⇒ Ecran SVGA

  ⇒ Souris ou autre périphérique de pointage

- **Configuration recommandée :**

  ⇒ PC Pentium® II

  ⇒ RAM 64 Mo

  ⇒ Disque dur (35 Mo disponibles)

  ⇒ Microsoft® Windows™ 2000 ou XP

  ⇒ Lecteur de CD-ROM (4X)

  ⇒ Ecran SVGA

  ⇒ Souris ou autre périphérique de pointage

Ce logiciel peut aussi fonctionner sous Microsoft® Windows NT™. Cette application ne travaille pas sous Unix, Mac, MS-DOS et Microsoft® Windows 3.11.

3-1-2- Procédure d'installation du logiciel DPL

Le logiciel Drive Programming Language est fourni sous forme de CD-ROM.
L'installation du logiciel se fait comme suit :

- Vérifier la configuration requise pour installer le logiciel

- Insérer le CD-ROM dans le lecteur approprié.
- Dans le menu déroulant Démarrer, sélectionner Exécuter... .
- Dans la boîte de dialogue « Exécuter », sélectionner Parcourir... .
- Dans la boîte de dialogue « Parcourir », sélectionner le lecteur où se situe le CD-ROM.
- Sélectionner Setup.exe puis Ouvrir dans la boîte de dialogue « Parcourir ».
- Sélectionner OK dans la boîte de dialogue « Exécuter ».

  Le programme d'installation du logiciel SPL débute.

- Le début de l'installation est marquée par une série de boîte de dialogue guidant l'utilisateur :
  1. répertoire de destination
  2. type d'installation (Typique, compacte ou personnalisée)
  3. sélection du dossier programme

Attention : seul un niveau de répertoire peut-être crée.

L'installation des fichiers débute et est indiquée par l'évolution d'une barre graphe.

L'installation se termine par l'ajout de l'icône du logiciel DPL dans le dossier programme.

3-2- Architecture du logiciel DPL

3-2-1- Les répertoires

Le logiciel DPL est installé par défaut dans le répertoire suivant :

C:\Program Files\SERAD\Dpl\

Il contient 4 sous répertoires :

- Data : contenant les sources du logiciel et la table de mots d'adressage par MobBus.
- Help : contenant l'aide du logiciel.
- Lib : contenant les différents fichiers de paramétrage du variateur.

- Os : contenant le système d'exploitation du variateur.
-

3-2-2- Contenu d'un projet

Un projet est composé d'un fichier .sdp et d'un répertoire du même nom.

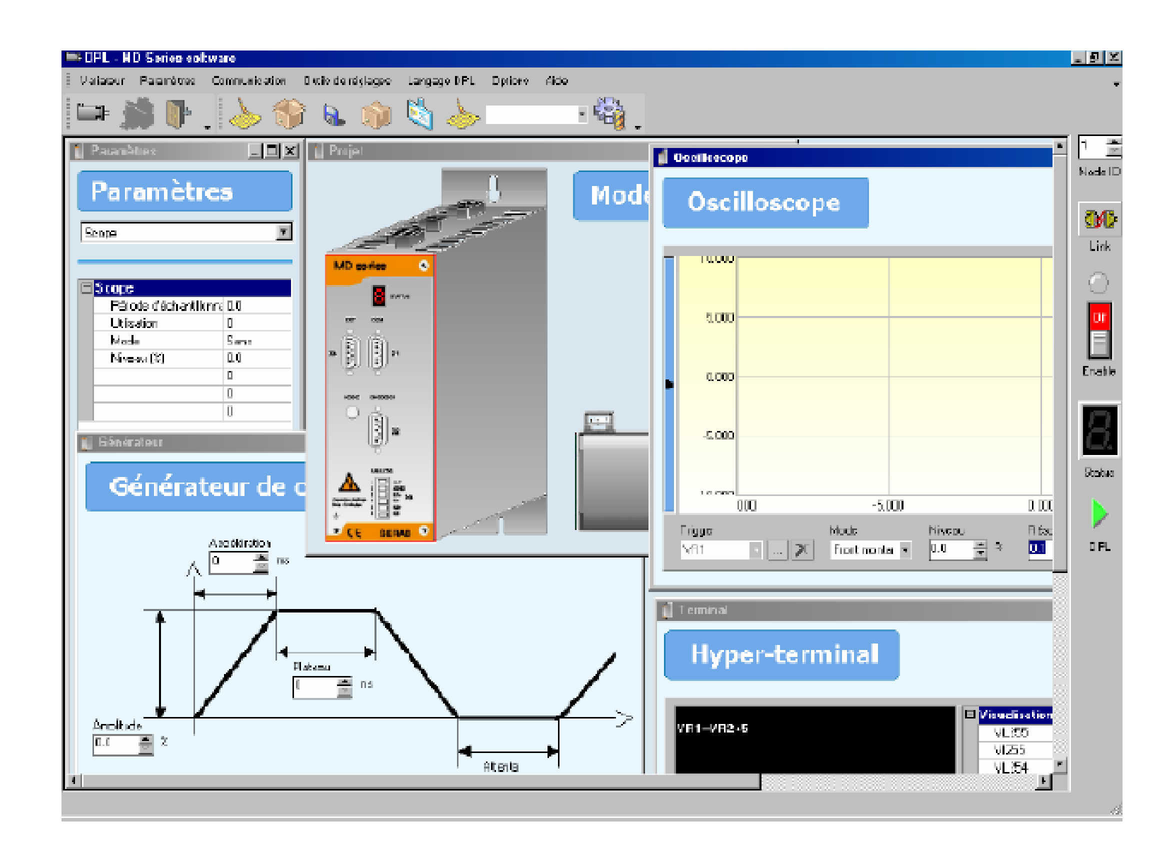
Dans ce répertoire, on y trouve :

- des fichiers .dpl contenant les différentes tâches sous format texte.
- un fichier .dpv contenant la liste des variables et leurs valeurs.
- Un fichier .dpi contenant les informations liées au projet.
- Un répertoire bin contenant les fichiers compilés des tâches et paramétrages nécessaires au variateur.

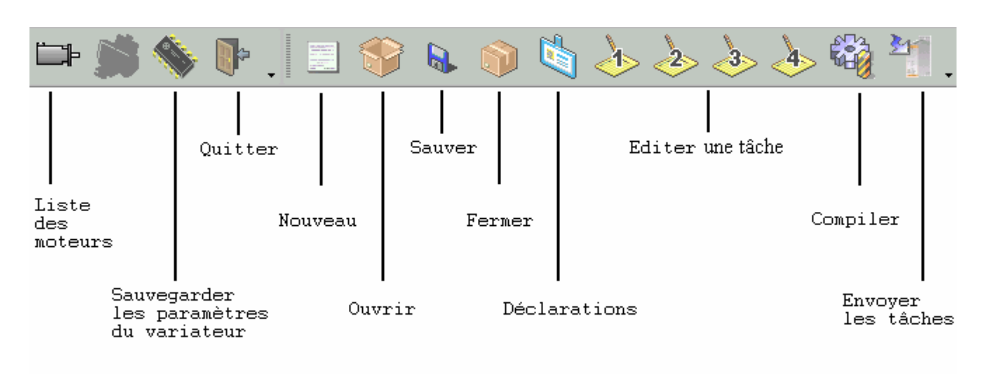
3-3- Présentation

3-3-1- Ecran initial

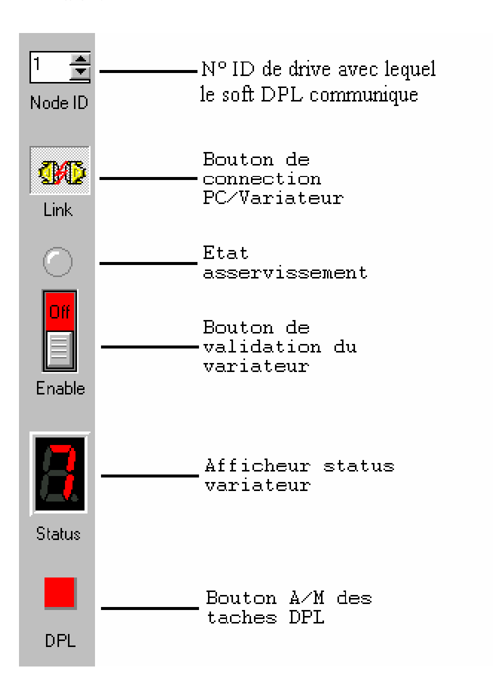
Le logiciel DPL est caractérisé par une fenêtre principale comportant le menu principal, une barre d'icônes et le multi-fenêtrage. Les propriétés du multi-fenêtrage permettent à l'utilisateur de pouvoir passer d'une fenêtre à une autre sans avoir à la modifier.



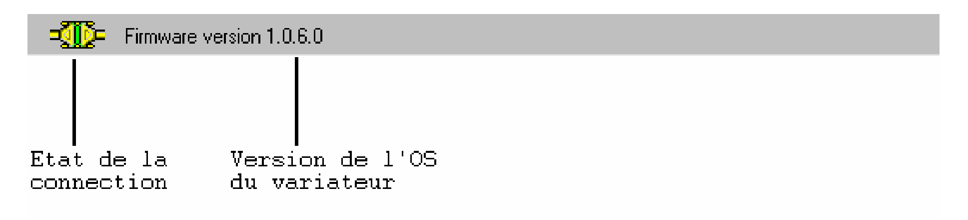
- La barre d'icônes :



- La barre de commande :



- La barre d'état :

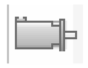


3-4- Menus et icônes

3-4-1- Variateur



- **Liste des moteurs :**

Icône : 

Action : Permet de charger les paramètres d'un moteur de la bibliothèque.

- **Préférences :**

Icône : 

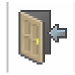
Action : Cette commande permet à l'utilisateur de définir son type d'impression (imprimante, papier, etc.…). L'orientation du papier peut-être modifiée mais n'est pas prise en compte lors de l'impression (impression de type portrait).

- **Imprimer :**

Icône : 

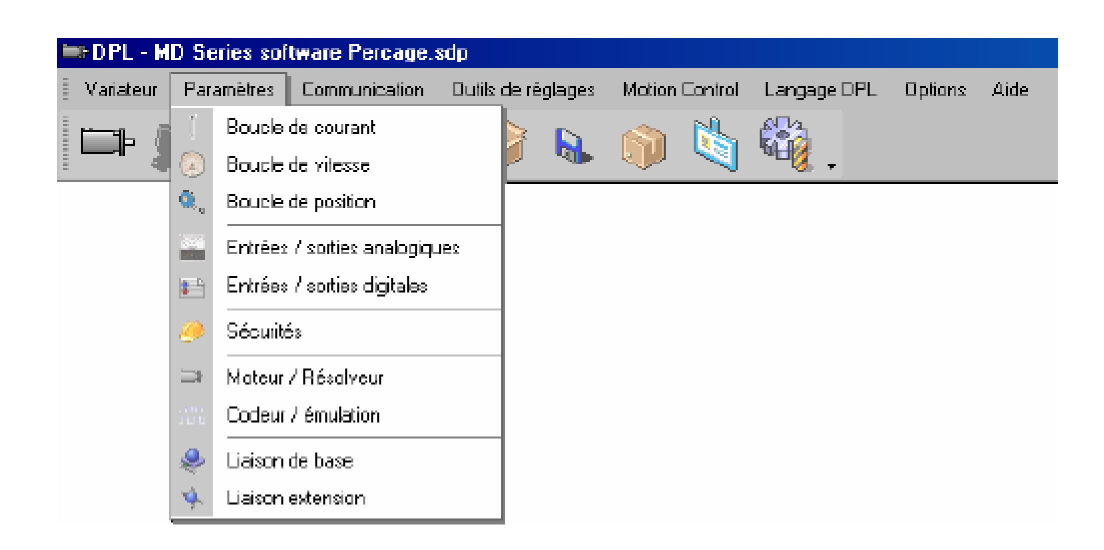
Action : Cette commande réalise une impression totale ou personnalisée du projet.

- **Quitter :**

Icône : 

Action : Cette commande permet à l'utilisateur de quitter le logiciel.

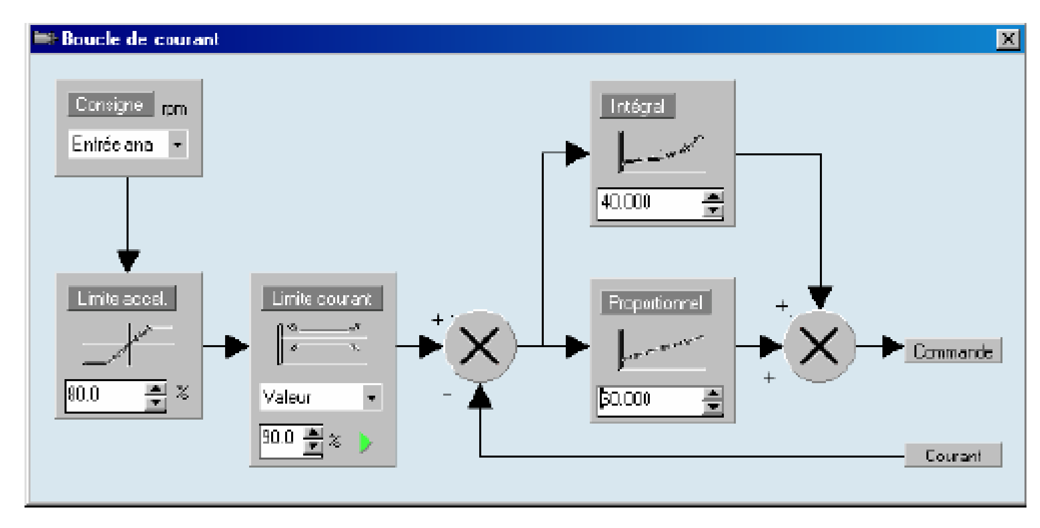
3-4-2- Paramètres



- **Boucle de courant :**

Icône :

Action : Permet de configurer la boucle courant du variateur



- Consigne : Sélection de la source (valeur, entrée analogique ou RS232) exprimée en pourcentage du courant maximal du moteur.
- Limite accélération : Limitation de la pente de la variation de courant
- Limite courant : Limitation du courant en pourcentage du courant maximal du moteur.
- Gain intégral : Régulation

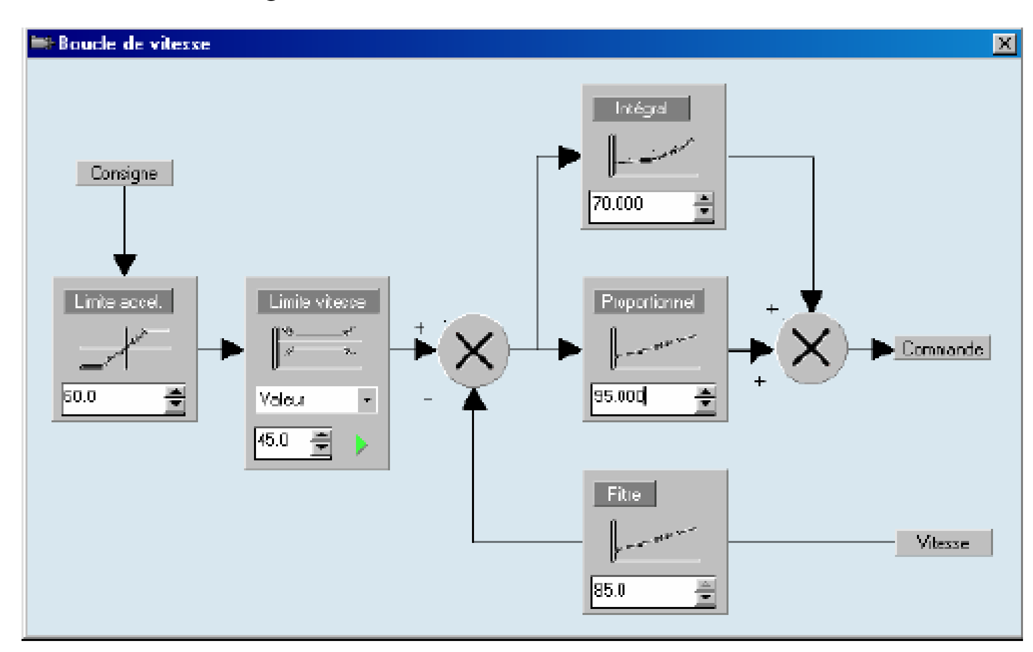
- Gain proportionnel : Régulation

Les limites d'accélération et de courant sont accessibles en mode *paramètres avancés* (voir Menu / Options / Accessibilité)

- **Boucle de vitesse :**

Icône : 

Action : Permet de configurer la boucle vitesse du variateur



- Consigne : Sélection de la source : valeur, entrée analogique, RS232 …
- Limite accélération : Limitation de la pente de vitesse
- Limite vitesse : Limitation de la vitesse en pourcentage de la vitesse nominal
- Gain intégral : Régulation
- Gain proportionnel : Régulation
- Filtre : Filtrage sur le calcul de la vitesse réelle.

Les limites d'accélération et de vitesse, ainsi que le filtre sont accessibles en mode *paramètres avancés* (voir Menu / Options / Accessibilité)

- **Boucle de position :**

Icône : 

Action : Permet de configurer la boucle de position du variateur.

Boucle de position

Anticipation
Vitesse théorique
35.000
Proportionel
Consigne
80.000
Vitesse
Intégral
90.000
Position

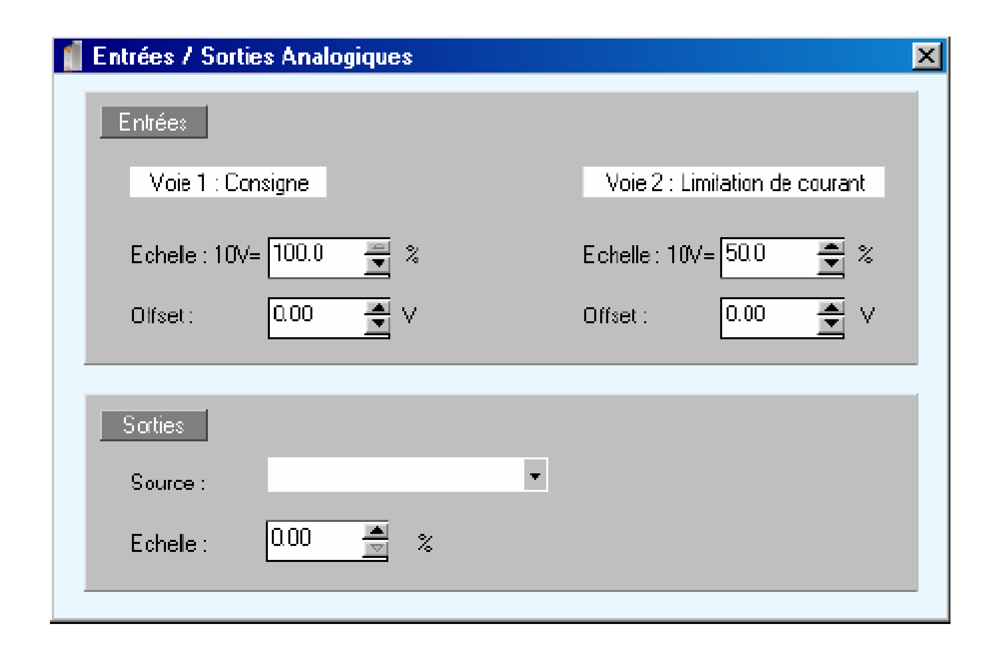
- Anticipation : Le gain d'anticipation de vitesse assure une erreur de poursuite proche de zéro.
- Gain proportionnel : Une valeur trop faible donne un asservissement mou, une valeur trop forte rend le système instable.

Le gain intégral est accessible en mode *paramètres avancés* (voir Menu / Options / Accessibilité)

- **Entrées/Sorties analogiques :**

Icône : 

Action : Permet de configurer les entrées et sorties analogiques.



- Echelle : Valeur en pourcentage du courant nominal (en mode couple), de la vitesse nominale (en mode vitesse) pour 10V sur l'entrée analogique.

  Ex : Vitesse nominal -> 3000tr/min

  Echelle -> 100%

  Variateur en mode vitesse

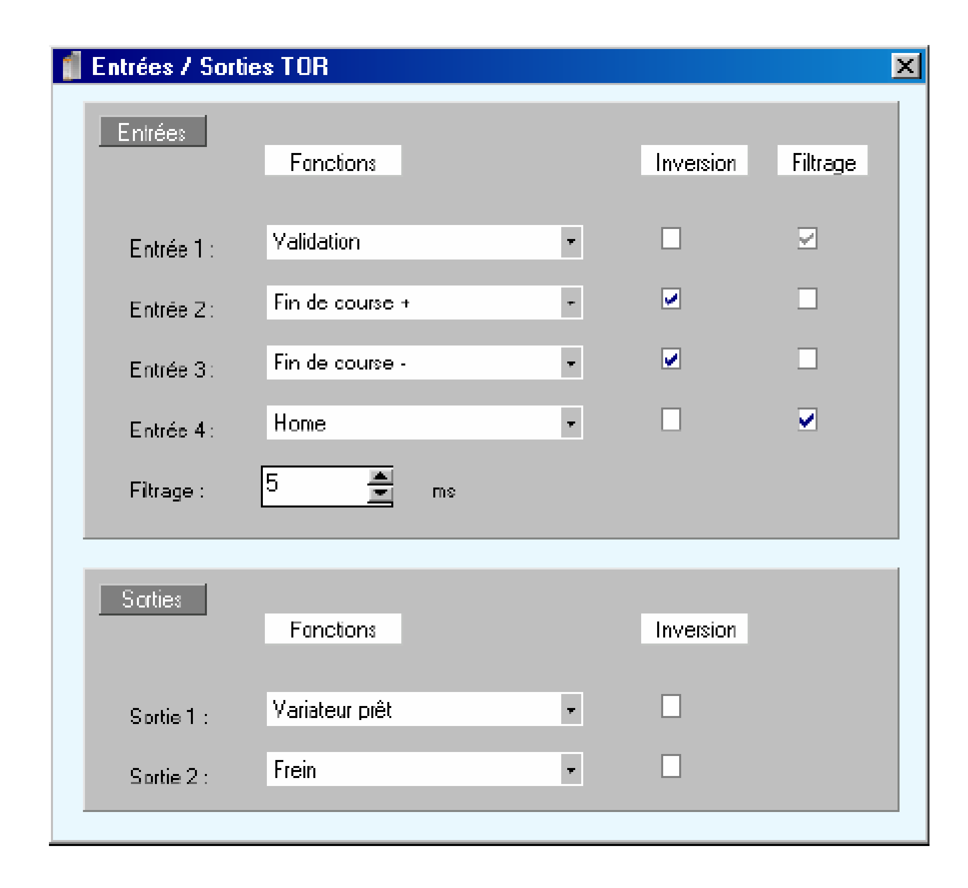
  Pour une tension de 10V sur l'entrée analogique, la consigne moteur sera de 3000 tr/min.

- Offset : Ajoute un offset à la valeur réelle reçue.

- **Entrées/sorties digitales :**

Icône :

Action : Permet de configurer les entrées et sorties logiques du variateur.

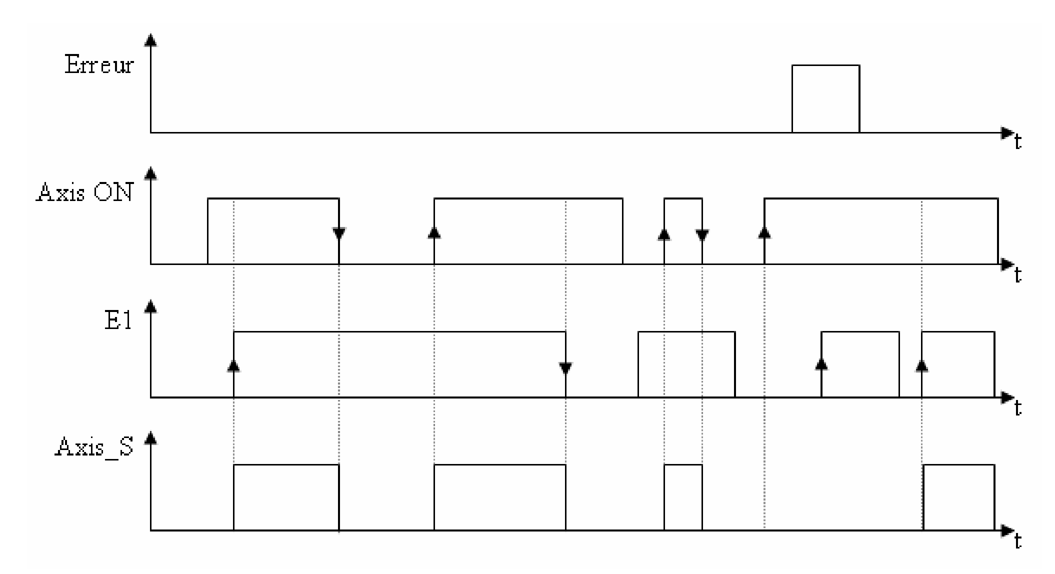


- Entrée 1 : Sélection : **Validation** du variateur ou aucune :

1. Si **Aucune**, l'asservissement se fait par le bouton enable de la fenêtre principale du DPL ou par l'instruction Axis on du langage DPL.

2. Si **Validation**, l'asservissement se fait sur front montant de l'entrée logique E1.

3. Si **Validation + iDPL**, la demande d'asservissement se fait par front montant sur 1 des 2 conditions et niveau logique 1 sur la seconde:



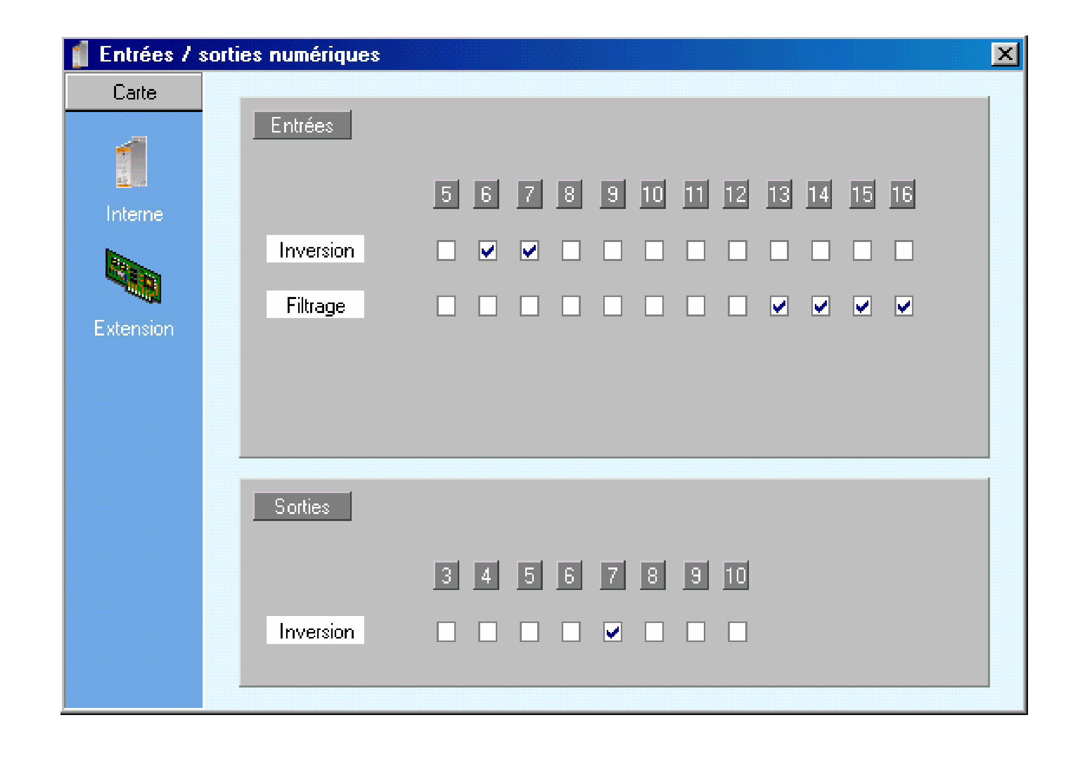
- Entrée 2 : Sélection : **Fin de course +** ou aucune.
- Entrée 3 : Sélection : **Fin de course -** ou aucune.
- Entrée 4 : Sélection : **Capteur prise d'origine**, **Raz défaut** sur front descendant ou aucune.
- Délai de Filtrage : Valeur du filtre en ms.
- Inversion : Si inversion non activée, l'entrée est gérée en logique positive sinon en logique négative.
- Filtrage : Permet d'activer le filtrage sur l'entrée sélectionnée.
- Sortie 1 : **Variateur prêt** ou aucune.
- Sortie 2 : **Frein moteur** ou aucune

La sortie variateur prêt doit être insérée dans la boucle d'arrêt d'urgence.

Si le frein est sélectionné sur la sortie 2, il est nécessaire d'insérer un relais statique externe (la sortie n°2 étant limitée à 100 mA).

L'état logique de la sortie frein correspond à l'état *enable* interne du variateur

La décélération urgente (motion control / profil de vitesse) est utilisée pour arrêter le mouvement lorsqu'on utilise les entrées de Fin de course lorsque le variateur est en mode position.



En ajoutant une carte d'extension I/O, vous disposez de :

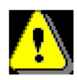
- 12 entrées supplémentaires pouvant être filtrées et/ou inversées.
- 8 sorties supplémentaires pouvant être inversées.

- **Sécurités :**

Icône : 

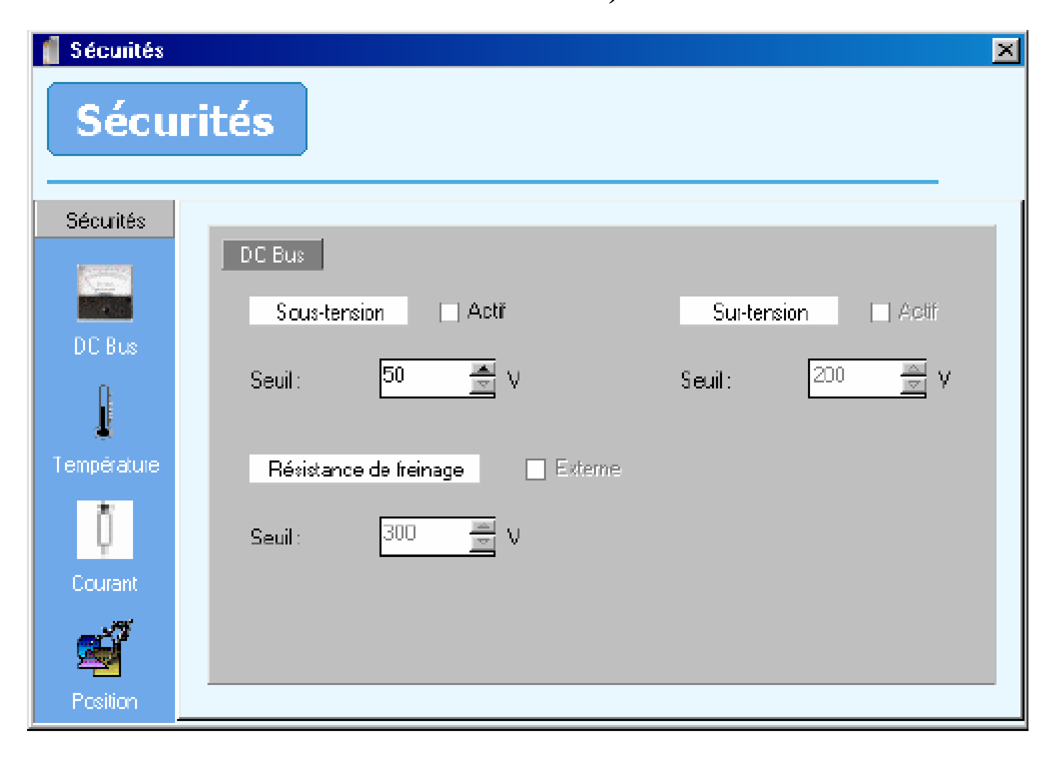
Action : Permet d'ajuster les paramètres de sécurité pour une sécurité maximale.

SECURITE DC BUS :

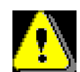
 **Réglages usines, ne pas les modifier.**

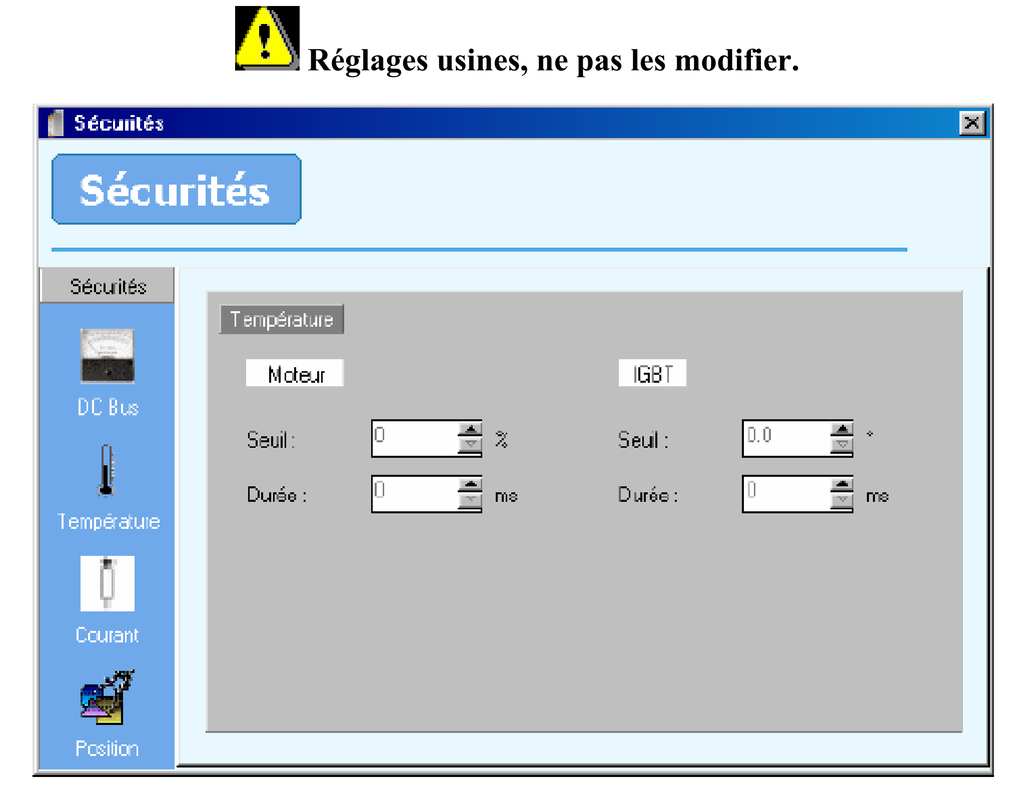
Dans le cas où une résistance externe est nécessaire, cocher la case Résistance externe.

⚠ **cette résistance aura dû être bien dimensionnée sous peine de détérioration de celle-ci, ce réglage est accessible en paramètres avancés (voir Menu /Options / Accessibilités).**

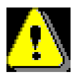


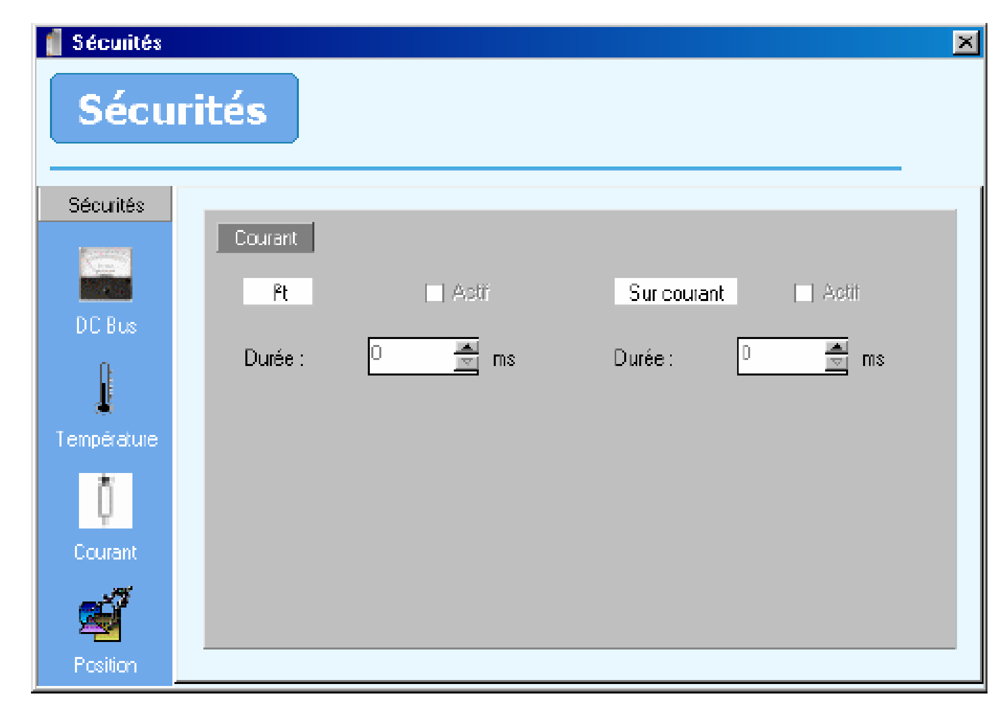
SECURITE TEMPERATURE :

 **Réglages usines, ne pas les modifier.**



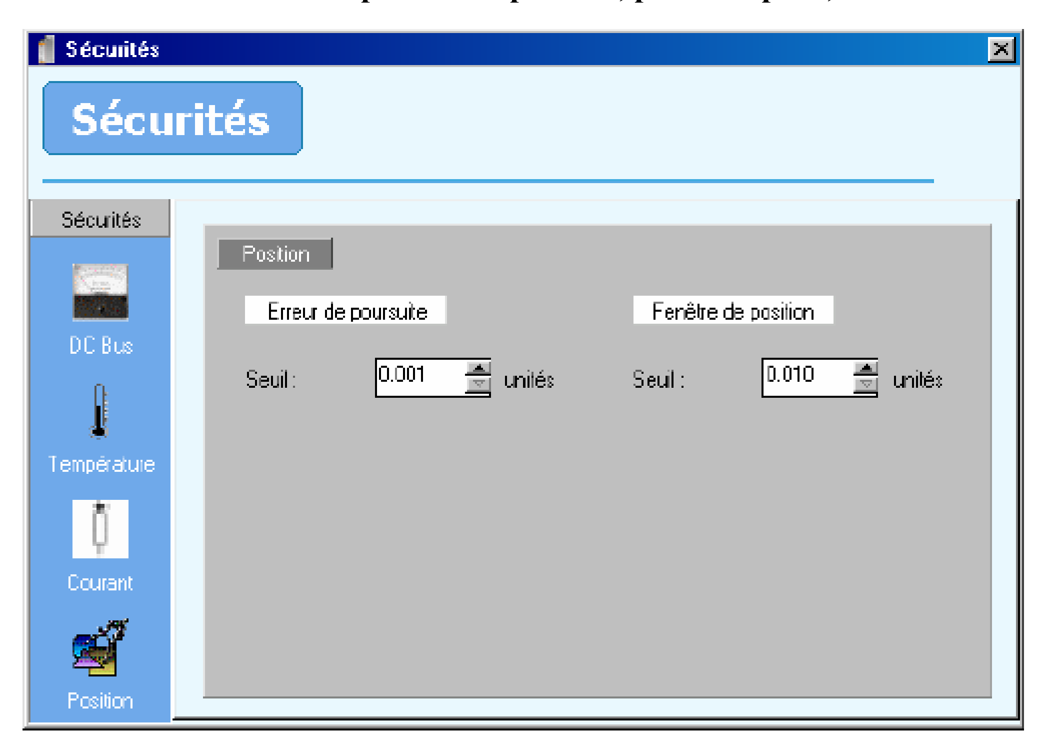
SECURITE COURANT :

 **Réglages usines, ne pas les modifier.**



SECURITE POSITION :

Lorsque le variateur est utilisé en mode position, régler le seuil d'erreur de poursuite au minimum. Attention, la valeur maximale admissible est de 8 tours moteurs. La valeur de ce seuil doit être la plus faible possible, par exemple 0,2 tour moteur.



- Erreur de poursuite : Dès que l'axe passe en mode asservi, il est contrôlé à tout moment : à l'arrêt, en mouvement. Si la différence entre sa position théorique calculée et sa position réelle donnée par le retour codeur est supérieure à l'erreur de poursuite maxi, le variateur passe l'axe servo en mode non asservi.

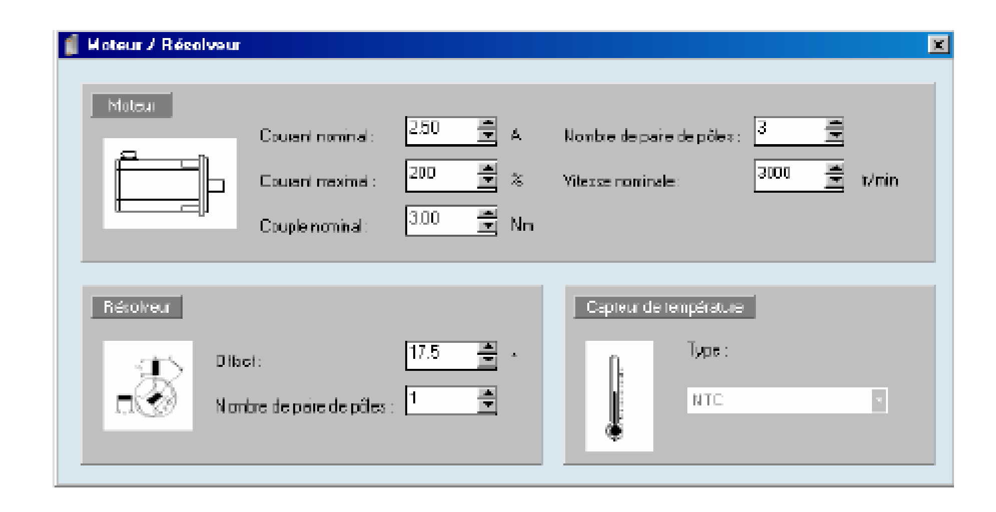
  Le réglage de cette valeur est très importante : une valeur trop petite entraîne des arrêts intempestifs sur l'axe, une valeur trop grande influe sur la sécurité des organes électriques et mécaniques.

- Fenêtre de position : Ce paramètre est utilisé pour modifier la fenêtre de positionnement minimale entre la position réelle et la position théorique. Après un déplacement, si la différence entre la position réelle et la position demandée est inférieure à la fenêtre de position, le système considère que la position est atteinte.

- **Moteur/Résolveur :**

Icône :

Action : Permet de configurer le moteur et le résolveur.



1. Moteur :

Courant nominal : Courant nominal du moteur en A.

Courant maximal : Pourcentage par rapport au courant nominal. Par défaut 200% (Imax = 2 * Inom).

Couple nominal : Couple nominal du moteur en Nm. Cette information n'est pas utilisée et est juste à titre indicatif.

Nombre de paire de pôles : Suivant le type de moteur.

2. Résolveur :

Offset : calage résolveur.

Nombre de paire de pôles : figé pour 1 paire de pôles

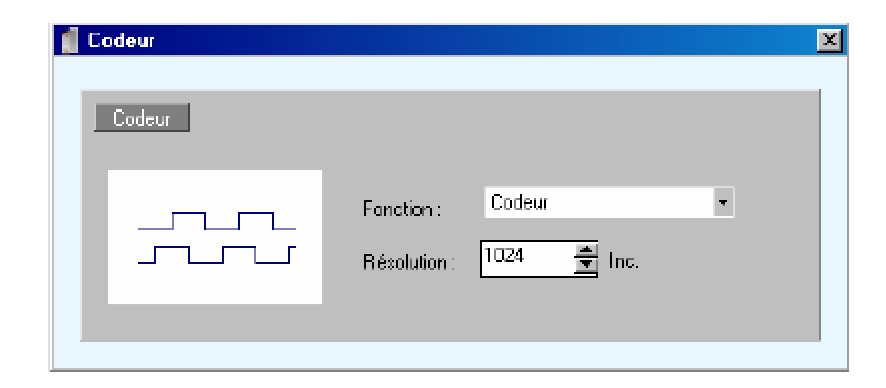
3. Capteur de température :

Type : Réglage usine (PTC ou NTC).

- **Codeur/Emulation :**

Icône :

Action : Permet de paramétrer le codeur

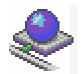


Fonction : Sélection : entrée codeur maître ou sortie émulation codeur

Résolution : Codeur maître : rentrer la résolution en nombre d'incréments (4 incréments par point). Ex : pour un codeur 500 points rentrer 2000 incréments.

Emulation codeur : possibilité de choisir entre 256, 512, 1024, 2048 ou 4096 incréments (4096 par défaut).

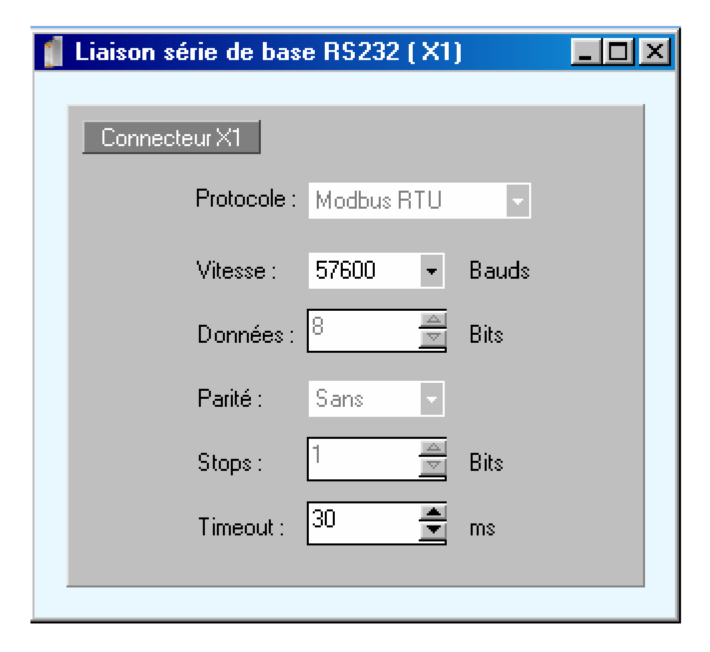
- **Liaison RS232 de base :**

Icône : 

Action : Permet de paramètrer la communication Modbus.

Le variateur gère cette liaison en Modbus RTU esclave.

Le format 8 bits de données, 1 bit de stop, pas de parité, est figé.



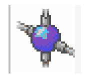
Dans cette fenêtre, on paramètre la vitesse de transmission et le timeout dans le cas où l'on est pas en « communication système ». Lorsque l'on utilise cette

liaison en « communication système » (réglage par défaut à partir du menu Options / ComPC), la vitesse est figée à 57600 bauds.

En communication système, le signal RTS du PC est utilisé et forcé à l'état logique 1.

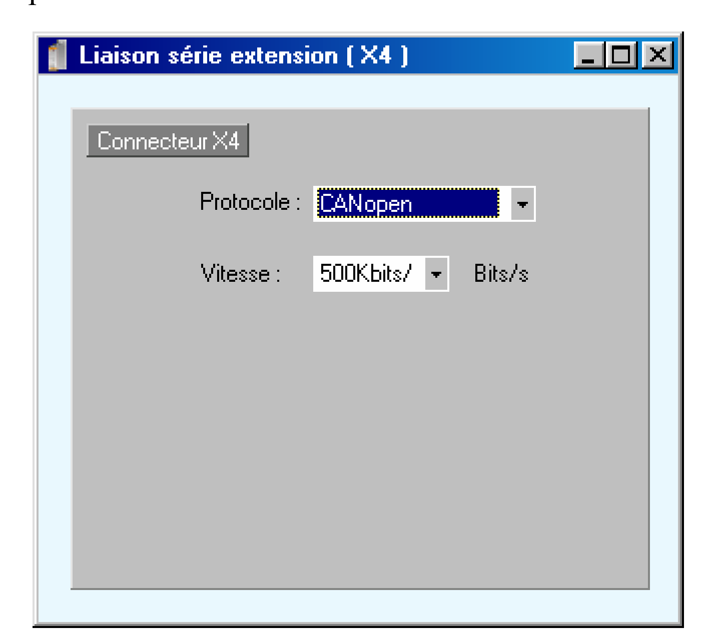
Les messages échangés sur cette liaison sont toujours adressés à l'esclave n°1 (n° esclave Modbus = 1).

- **Liaison d'extension :**

Icône : 

Action : Permet de paramètrer la liaison d'extension en CANopen, RS232, RS422 ou RS485.

- CANopen :



Vitesse : permet de définir la vitesse de communication sur le bus CANOpen.

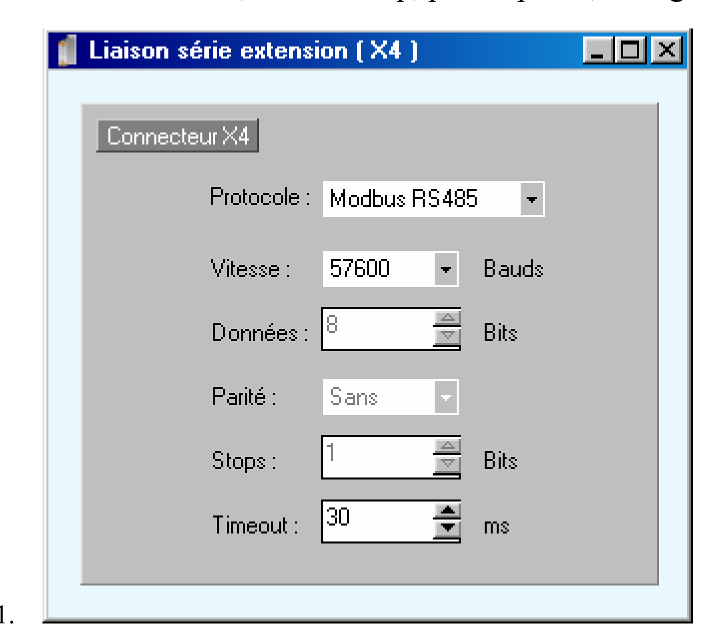
Pour plus de renseignements, voir les annexes sur le bus CANOpen .

Choisir un numéro de Node ID dans l'écran principal pour communiquer avec le variateur associé sur le bus CANopen.

- Port RS232, RS422 ou RS485 :

Le variateur gère cette liaison en Modbus RTU esclave.

Le format 8 bits de données, 1 bit de stop, pas de parité, est figé.

1. 

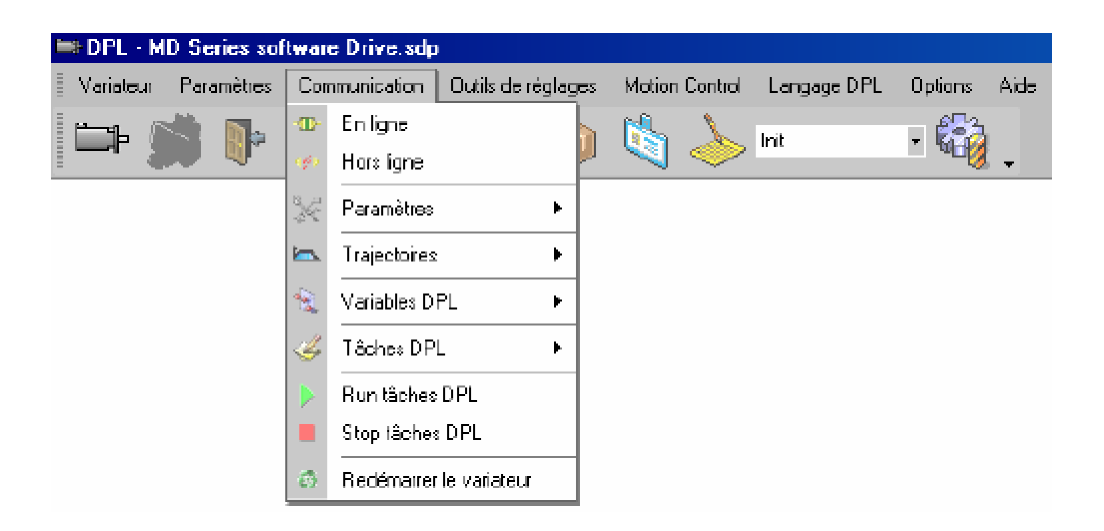
- Protocole : permet de choisir le support de la liaison.

  Le numéro de NodeID (adresse) du variateur correspond à la roue codeuse en face avant du variateur.

  NodeID = Position roue codeuse + 1. Exemple : position roue codeuse sur 5, NodeID vaut 6.

- Vitesse : permet de définir la vitesse de communication du port.
- TimeOut : temps maximum de non réponse.

3-4-3- Communication

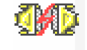


- **En ligne :**

Icône : 

Action : Permet d'établir la liaison avec le variateur. Tous les paramètres affichés sur l'écran du PC correspondent aux valeurs stockées dans le variateur.

- **Hors ligne :**

Icône : 

Action : Permet de travailler sans être relié au variateur.

- **Paramètres :**

Icône : 

Action : Si votre variateur communique avec le logiciel, vous pourrez :

- ***Envoyer les paramètres PC -> Variateur*** : permet d'envoyer un fichier de paramètres du PC vers le variateur. Ces paramètres sont automatiquement sauvés dans le variateur.
- ***Recevoir les paramètres PC <- Variateur*** : permet de recevoir dans un fichier sur le PC, les paramètres contenus dans le variateur.
- ***Sauvegarder les paramètres variateur*** : permet d'enregistrer les paramètres courants du variateur dans sa mémoire Flash pour en assurer la sauvegarde même après coupure.

Si votre variateur ne communique pas avec le logiciel, vous pourrez :

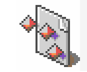
- ***Ouvrir un fichier paramètres*** : permet d'ouvrir un fichier paramètre du PC
- ***Enregistrer les paramètres dans un fichier*** : permet d'enregistrer les paramètres en cours dans un fichier paramètres.

- **Trajectoires :**

Icône : 

Action : Permet d'envoyer ou de recevoir les 64 profils de trajectoires préenregistrés.

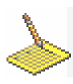
- **Variables DPL :**

Icône : 

Action : Permet d'envoyer ou de recevoir les variables initialisables du variateur.

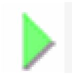
Seules les variables VR0 àVR63 et VL0 à VL63 sont concernées. A chaque mise sous tension du variateur, ces 128 variables sont chargées avec leur valeur initiale

- **Tâches DPL :**

Icône : 

Action : Permet d'envoyer ou d'effacer les tâches du variateur.

- **Run DPL :**

Icône : 

Action : Permet de démarrer le DPL. Le variateur exécute toutes les tâches activées et ayant un démarrage automatique.

- **Stop DPL :**

Icône : 

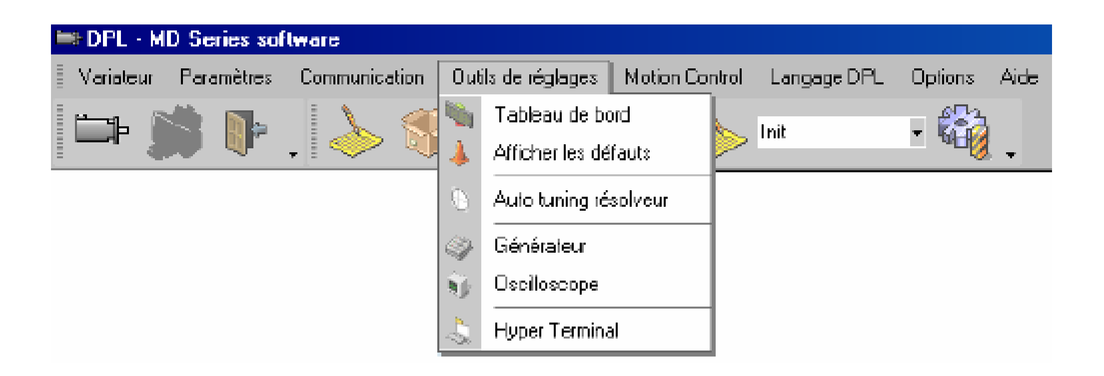
Action : Permet d'arrêter le DPL. Toutes les tâches s'arrêtent.

- **Redémarrer :**

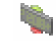
Icône : 

Action : Permet de redémarrer le variateur.

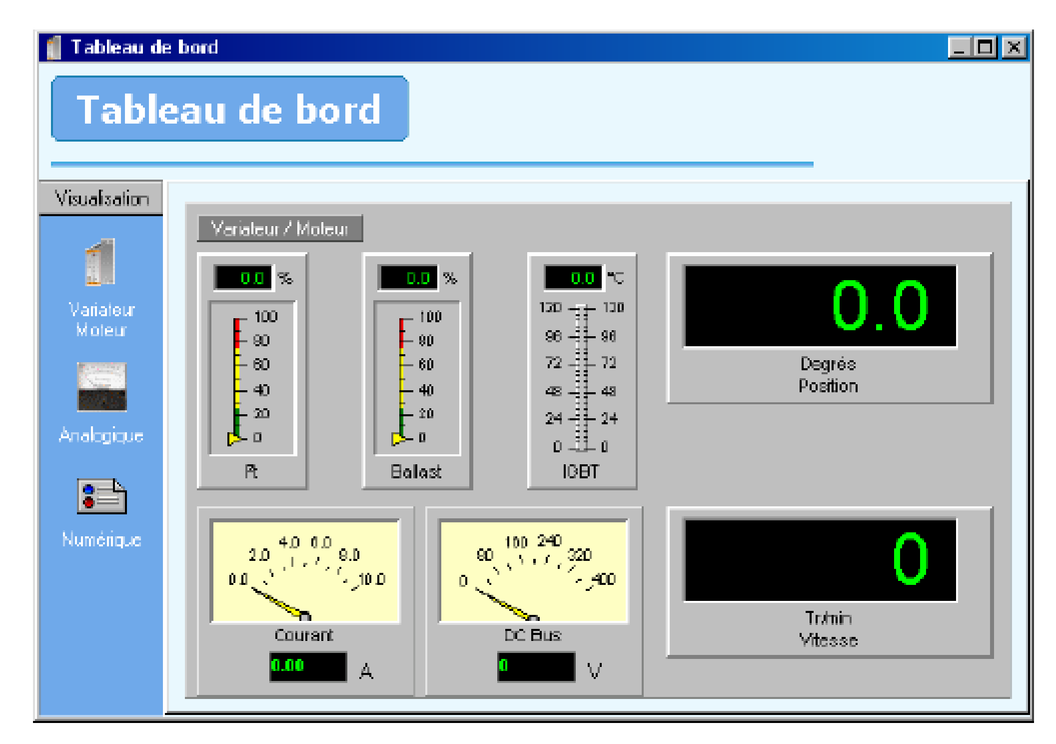
3-4-4- Outils de réglages



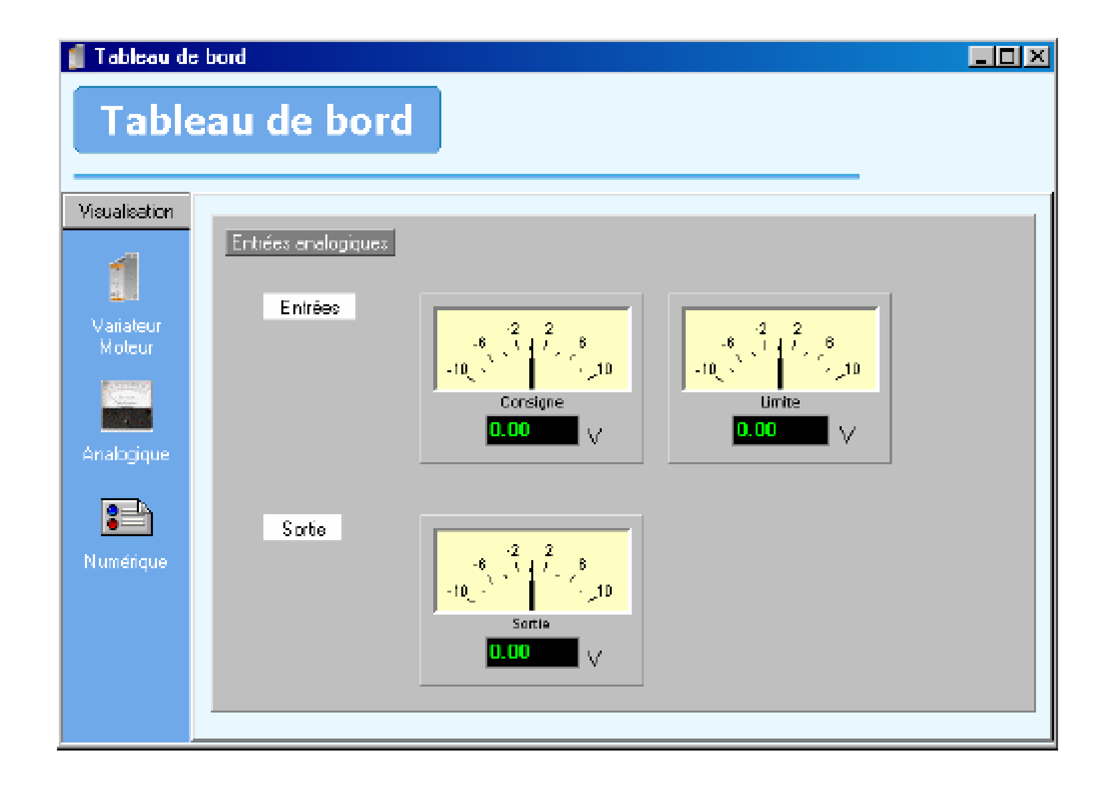
- **Tableau de bord :**

Icône : 

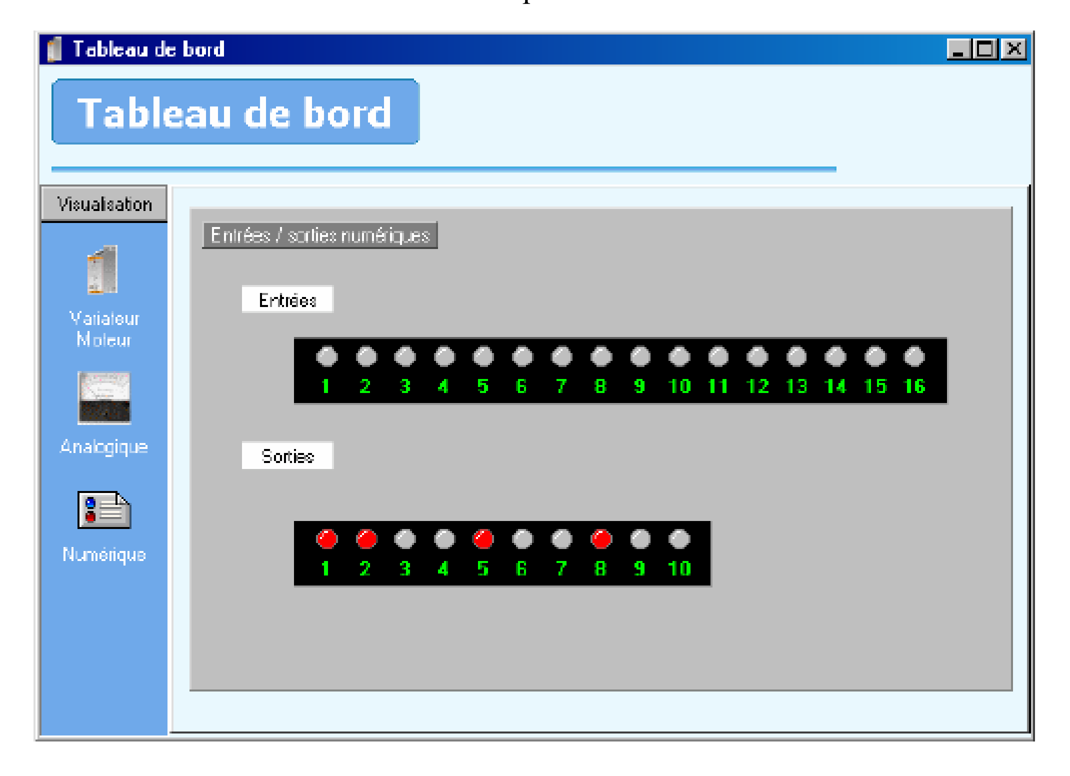
Action : Permet de visualiser l'état du variateur et du moteur :



Permet de visualiser l'état des E/S analogiques et modifier la sortie :



Permet de visualiser l'état des E/S numériques et modifier les sorties :



- **Afficher les défauts :**

Icône :

Action : Permet de visualiser les défauts du variateur

En cas de défaut, une dévalidation et revalidation du variateur (entrée E1 ou bouton enable dans l'écran principal du logiciel ou par l'instruction Axis off / Axis on du langage DPL) efface les défauts.

- **Auto tuning résolveur:**

Icône :

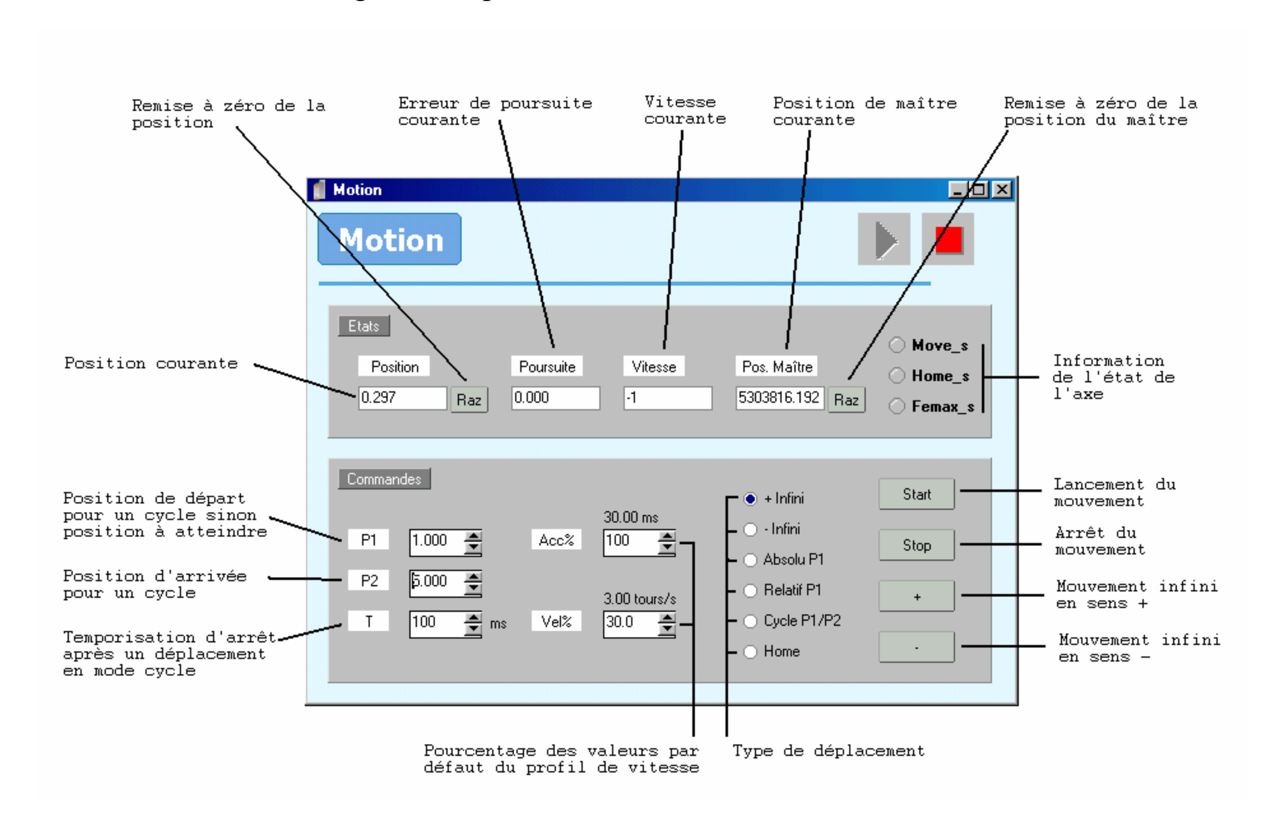
Action : Réalise un calage automatique entre le résolveur et le moteur.

Option disponible seulement si les paramètres avancés sont activés

- **Motion :**

Icône :

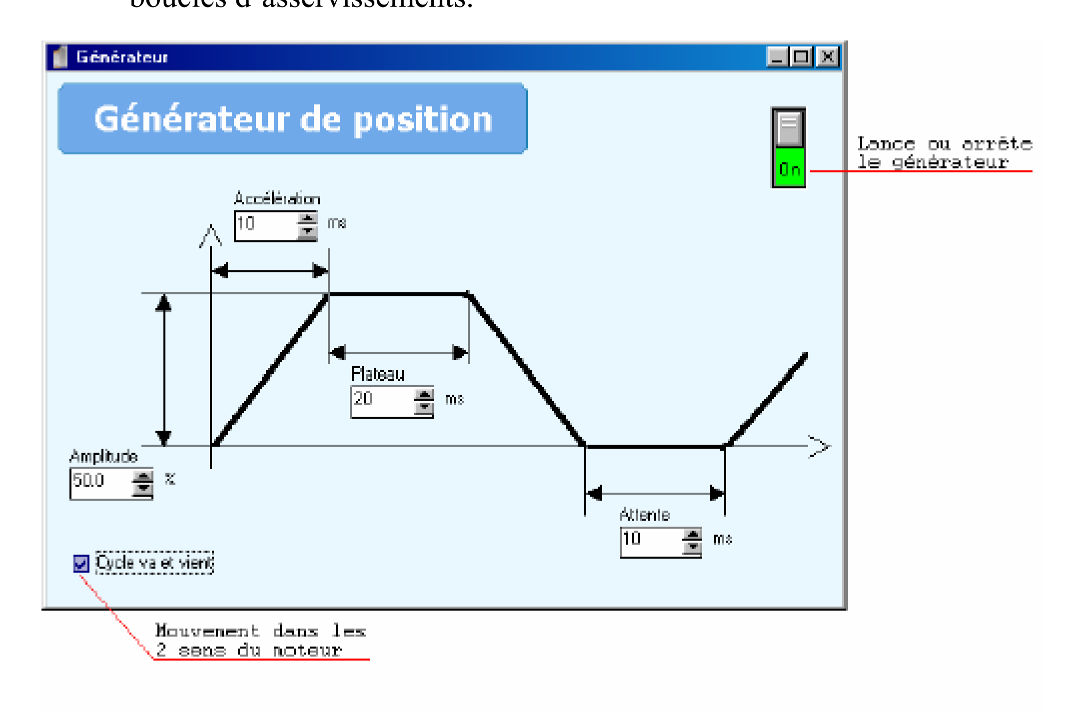
Action : Permet de tester la boucle de positionnement de l'axe. Il est préférable de commencer par vérifier le comportement du moteur/variateur en forçant la consigne à une valeur comprise entre +10V et –10V (L'axe doit être en mode débrayé). On peut ensuite passer en mode asservi et régler les paramètres d'asservissement. Si l'on souhaite sauvegarder ces modifications, il faut faire une sauvegarde des paramètres dans le variateur.



- **Générateur :**

Icône : 

Action : Permet de lancer différents types de trajectoires pour optimiser les tests des boucles d'asservissements.



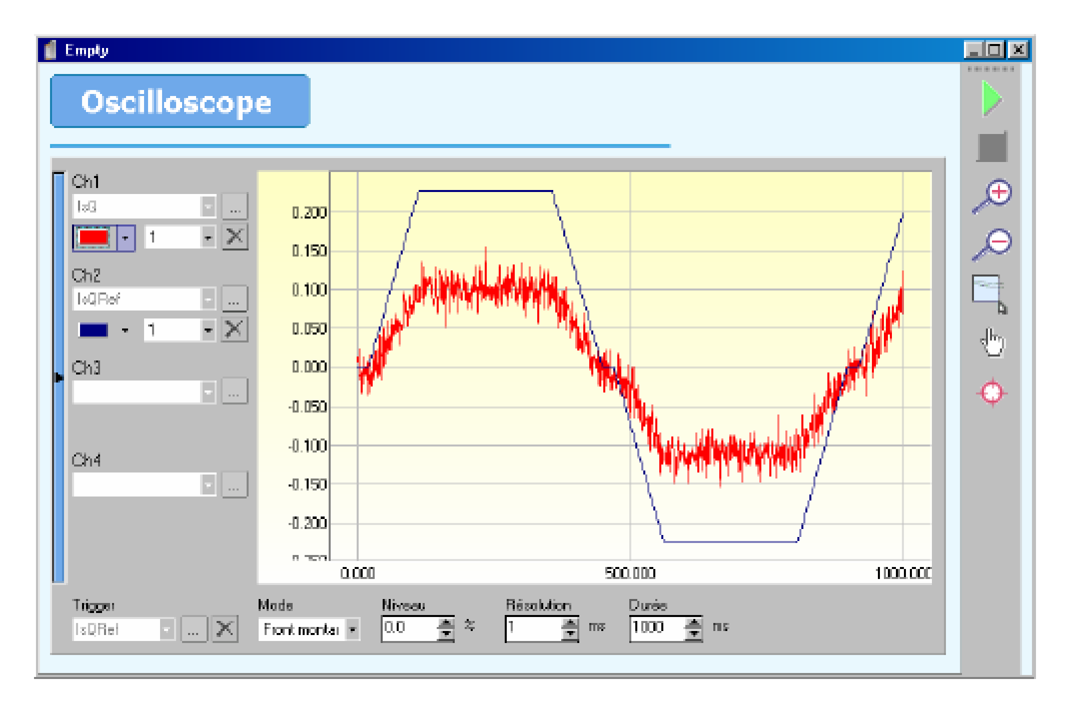
- Configurer le générateur pour effectuer le mouvement désiré.
- Asservir le moteur avec le bouton ENABLE (et/ou l'entrée E1 validation variateur)
- Lancer le mouvement avec le bouton ON/OFF du générateur

- **Oscilloscope :**

Icône : 

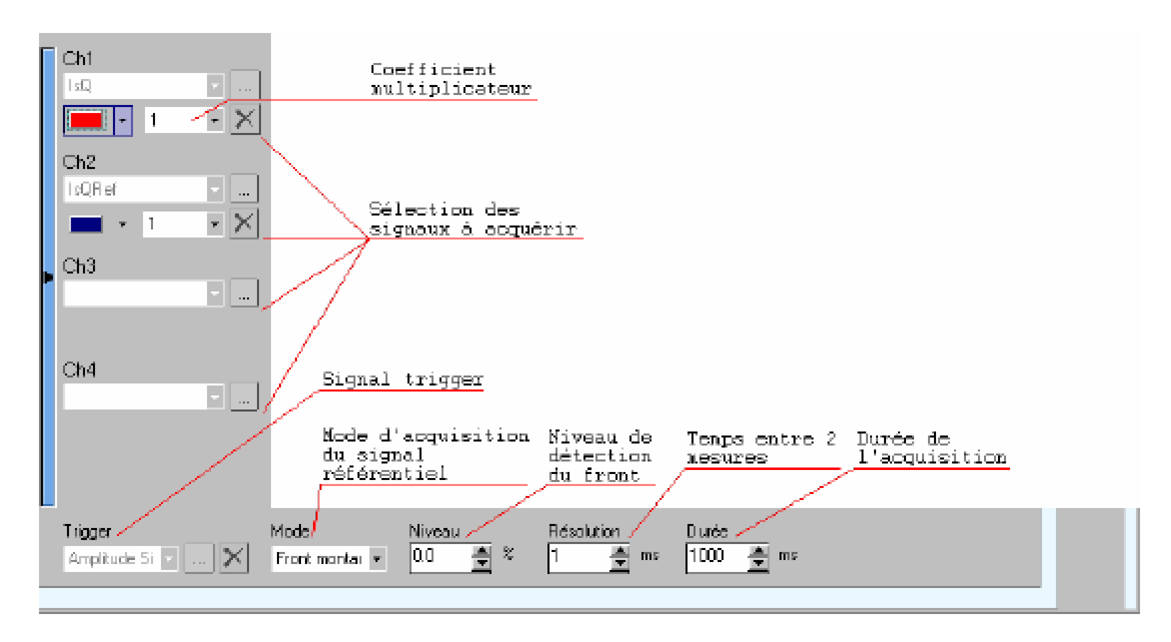
Action : Cette commande ouvre l'oscilloscope. Cet outil d'aide à la mise en œuvre permet de visualiser toutes les informations du variateur. Il est capable d'enregistrer jusqu'à 4 signaux simultanément.

L'oscilloscope est configuré en trois parties : l'écran de visualisation, la zone de configuration de l'acquisition, zone de réglage de la visualisation.



↳L'écran de visualisation est la partie centrale de l'oscilloscope où sont affichées les courbes.

↳La zone de configuration de l'acquisition permet de choisir les signaux à acquérir et de configurer le mode d'acquisition : le nombre d'échantillon, durée …

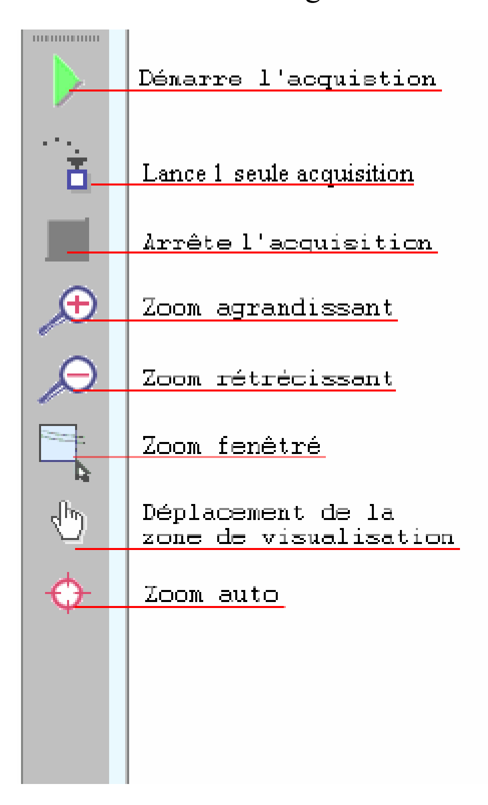


Chaque signal est affiché dans son unité

Exemple : courant en ampère, vitesse en tours/minute

Le coefficient multiplicateur d'un canal permet d'augmenter ou réduire l'amplitude du signal.

↳La zone de réglage de la visualisation permet de lancer ou arrêter l'acquisition et de modifier l'affichage de l'écran de visualisation.

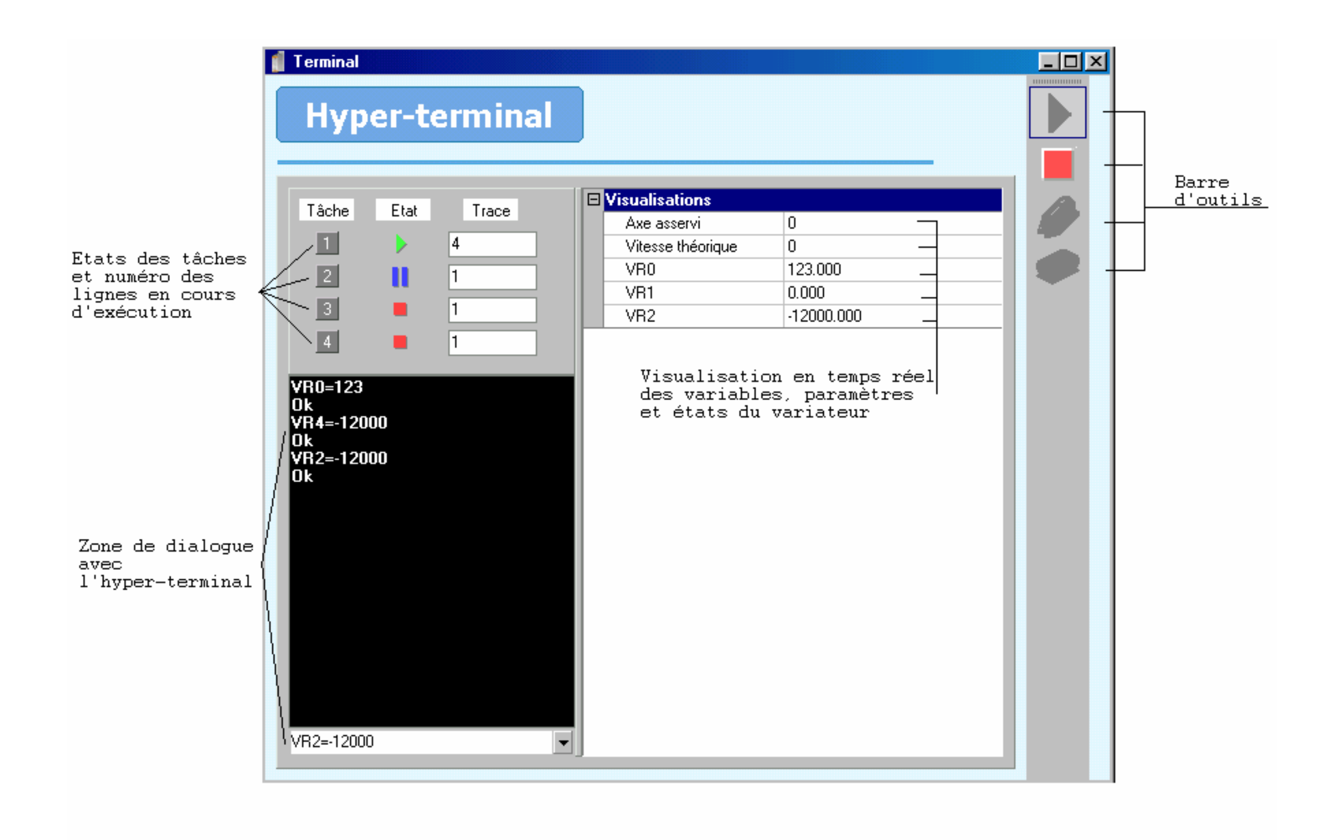


- Zoom fenêtré : Cliquer sur le bouton zoom fenêtré, le bouton devient actif puis tracer un rectangle dans le zone de visualisation des courbes en restant appuyer sur le bouton gauche de la souris, relâcher le bouton gauche pour valider le zoom.

- **Hyper terminal :**

Icône : 

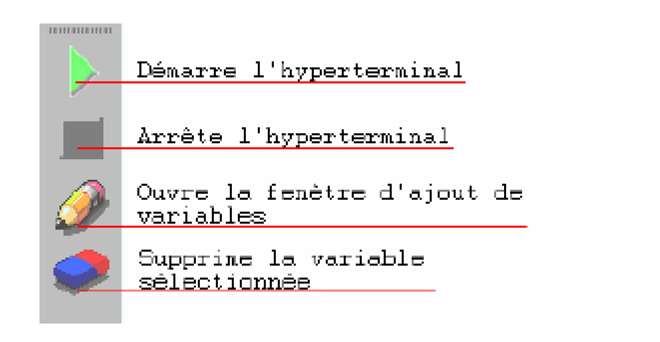
Action : Cette commande ouvre l'hyper terminal. Cet outil d'aide à la mise en œuvre permet d'interroger l'état du variateur, de visualiser et de modifier les variables, les paramètres, les entrées et les sorties.



↳ L'écran principal de la fenêtre Terminal permet la lecture et l'écriture de toutes les variables et paramètres en temps réel dans le variateur. Pour accéder à ces différentes informations, le terminal est muni de fonctions :

↳ <Nom de variable ou paramètre>=<Valeur> : affectation d'une valeur à une variable ou un paramètre

Pour faciliter l'édition des variables ou paramètres, un éditeur des propriétés de la configuration du variateur est mis à disposition. Cette fenêtre regroupe les différents paramètres et les différentes variables. En double cliquant sur une variable ou un des paramètres de cette fenêtre, le nom associé apparaît alors dans l'écran du terminal.



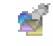
↳La fenêtre “ observations ” permet la visualisation de paramètres ou de variables en continue. Le nombre de variables ou paramètres à visualiser est limité à 16. Deux commandes permettent d'ajouter ou de supprimer une variable.

3-4-5- Motion control

Menu disponible seulement en mode position

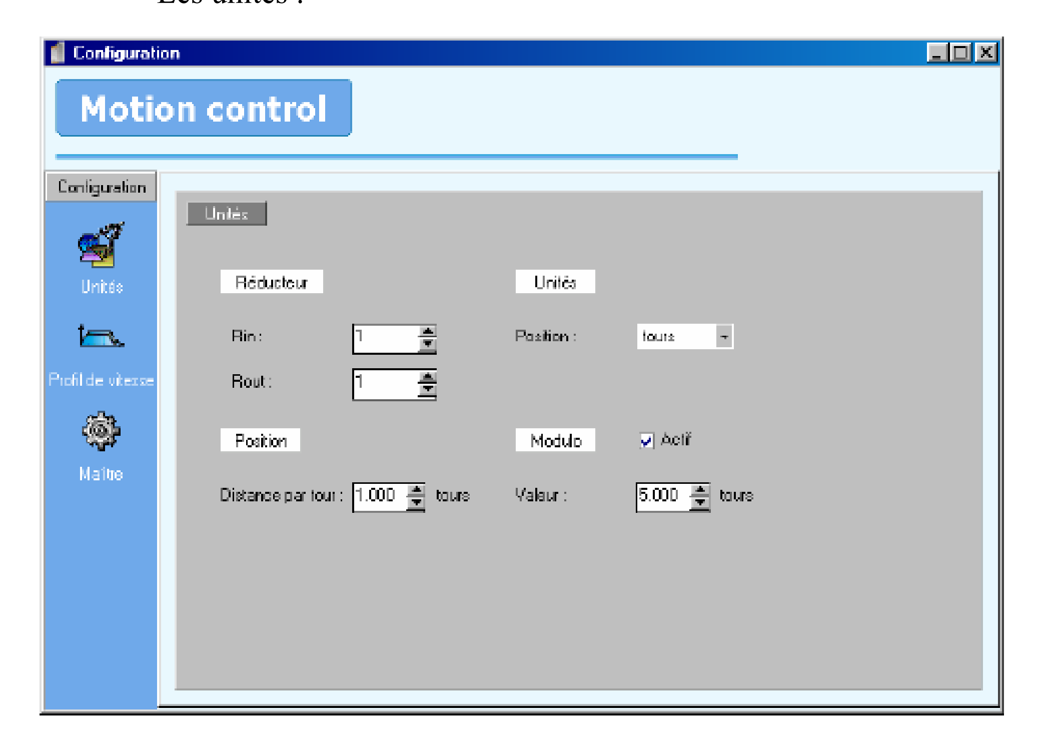


- **Configuration :**

Icône : 

Action : Permet de rentrer l'unité de travail (mm, degré …) ainsi que les vitesses, accélération et décélération par défaut.

- Les unités :



Exemple 1 : Axe infini

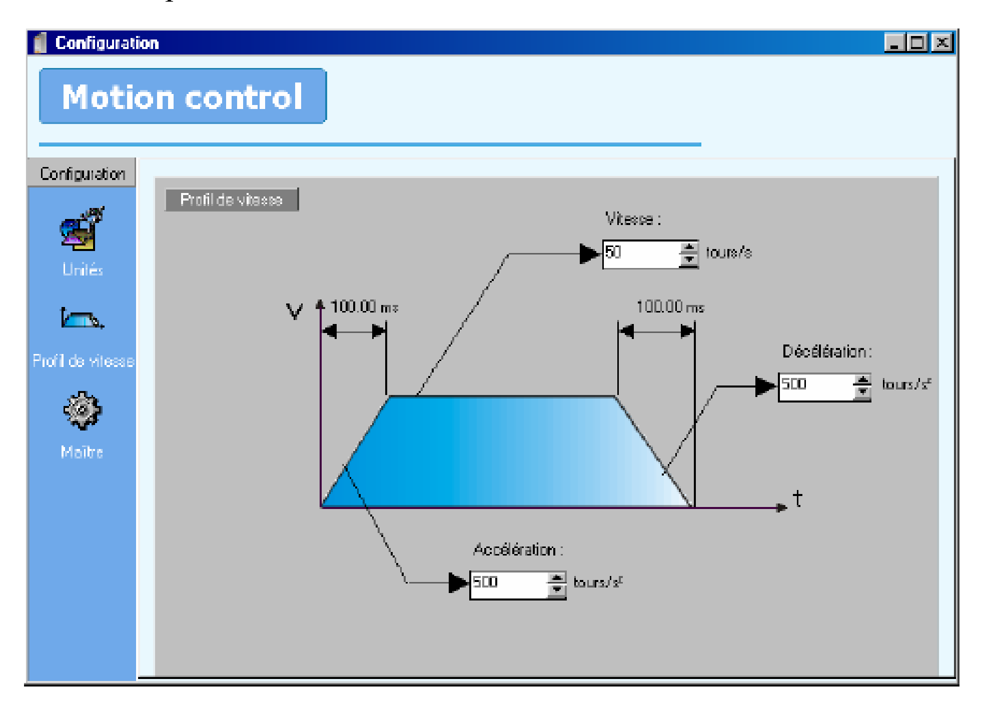
Moteur en bout de vis à bille au pas de 5mm. Unités = mm, Rin = 1, Rout = 1, Distance par tour = 5.000, Modulo non activé

Exemple 2 : Axe infini

Moteur avec réducteur de 10. En sortie de réducteur, tourelle 360°, Unités = degrés, Rin = 10, Rout = 1, Distance par tour = 360.000, modulo activé avec une valeur de 360.000

Nota : le nombre de chiffres après la virgule est paramétrable dans le menu *Options / Langage DPL*

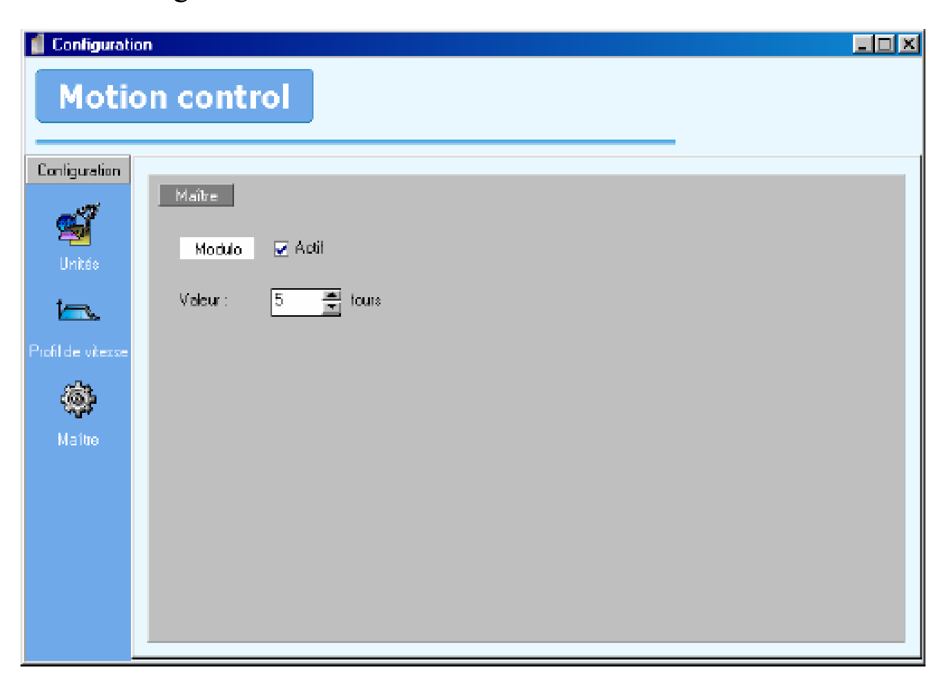
- Le profil de vitesse :



Les vitesses, accélérations, décélérations exprimées en pourcentage dans le générateur, dans les trajectoires, dans les instructions ACC%, DEC% du langage DPL font référence à ses valeurs

La décélération urgente est utilisée pour arrêter le mouvement lorsqu'on utilise les entrées de Fin de course.

- La configuration du codeur maître :

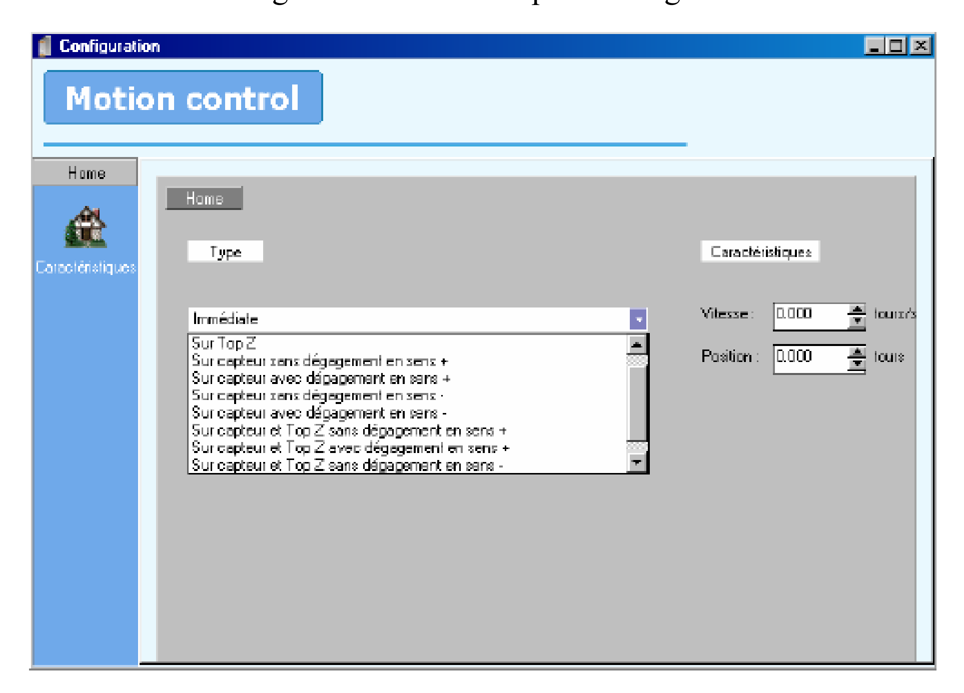


Le codeur maître utilise les mêmes unités que celle de l'axe moteur. Seul le modulo peut être différent.

- **Home :**

Icône : 

Action : Permet de configurer la fonction de prise d'origine de l'axe.



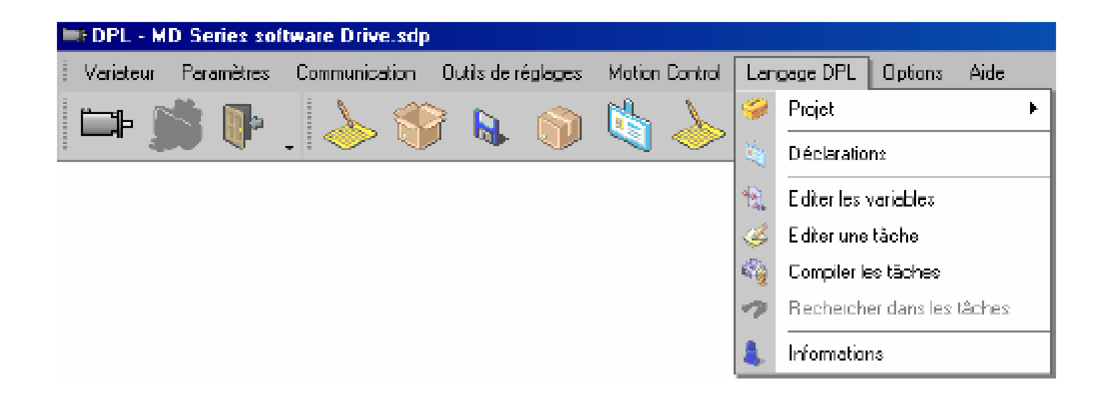
- Sélectionner le type de prise d'origine
- Saisir la vitesse à laquelle sera effectué le cycle d'origine.
- Saisir la position à charger dans le compteur lors de la détection de l'origine (par défaut 0)

- **Trajectoires :**

Action : Permet de lancer des trajectoires via les entrées du variateur.

Voir chapitre sur les trajectoires pré-enregistrées.

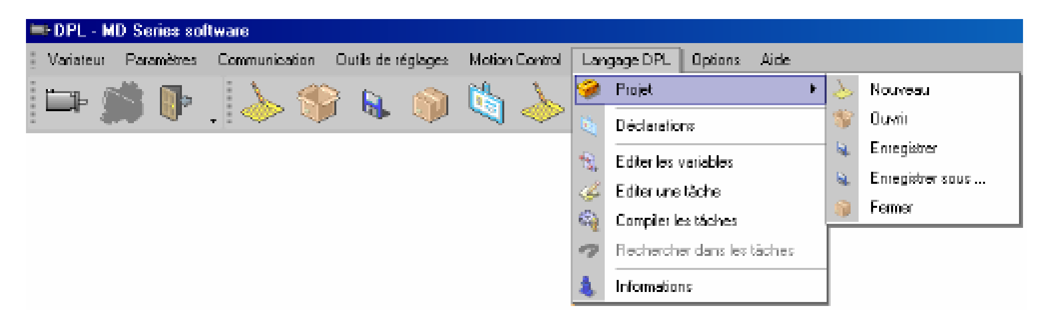
3-4-6- Langage DPL



- **Projet :**


Icône : 

Action : Permet d'accéder au menu projet du logiciel.



1. Nouveau :

Icône : 

Action : Cette commande permet à l'utilisateur de définir un nouveau projet.

2. Ouvrir :

Icône : 

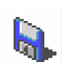
Action : Cette commande ouvre la boîte de dialogue "Ouvrir un Projet". Elle permet à l'utilisateur de spécifier le chemin et le nom du projet à charger.

3. Enregistrer :

Icône : 

Action : Cette commande permet la sauvegarde complète du projet en cours sous le nom spécifié.

4. Enregistrer sous :

Icône : 

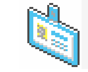
Action : Cette commande ouvre la boîte de dialogue "Enregistrer sous" et permet à l'utilisateur de spécifier le nom du projet de sauvegarde. Cette commande a pour effet de créer un fichier et un répertoire portant le nom spécifié avec pour le premier l'extension "sdp" et pour le second l'extension "data".

5. Fermer :

Icône : 

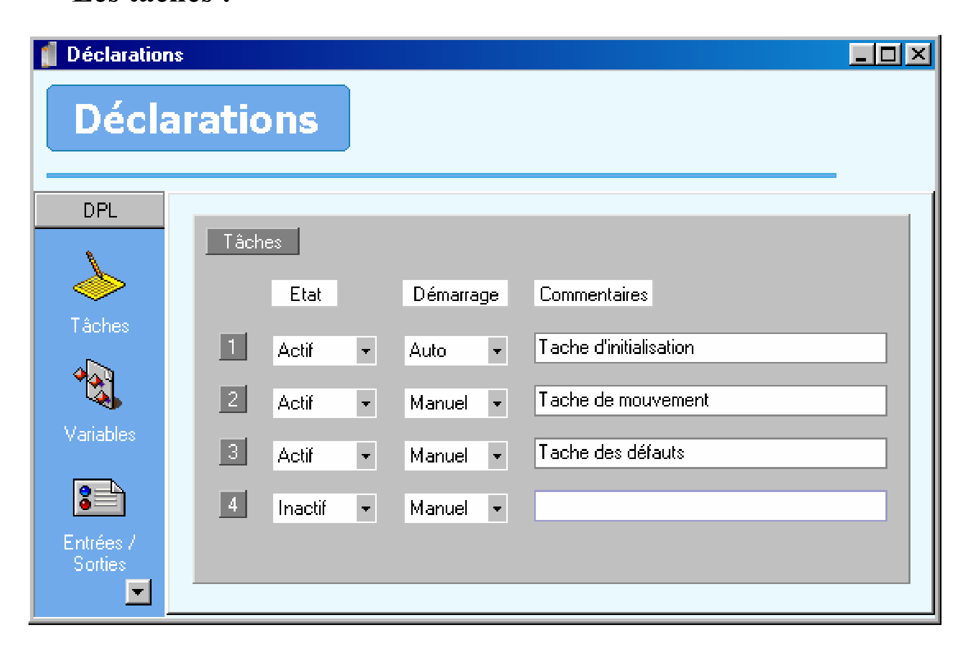
Action : Cette commande ferme le projet en cours.

- Déclarations :

Icône : 

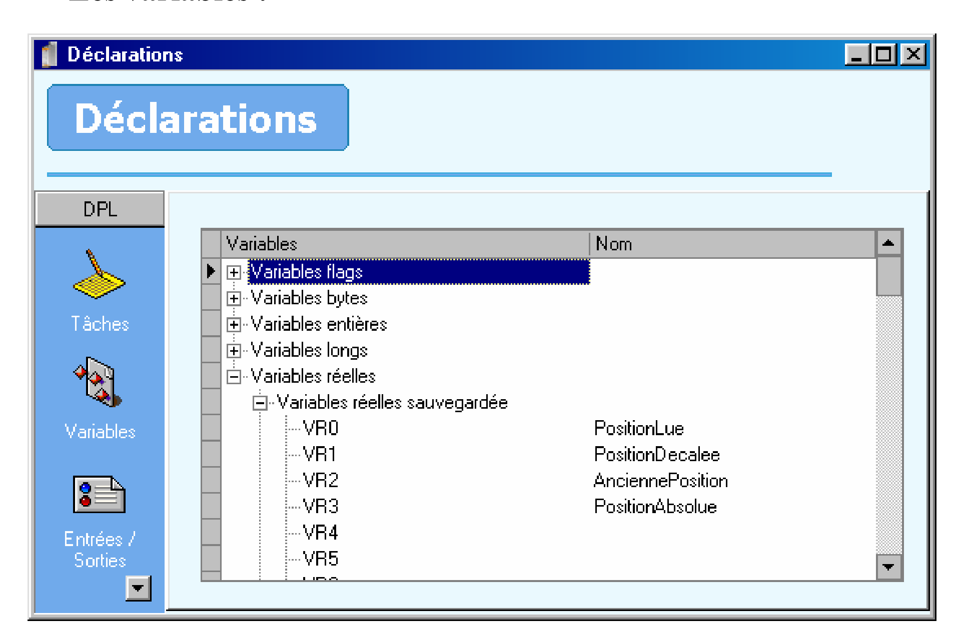
Action : Permet de déclarer les tâches, les noms de variables et les noms des E/S.

Les tâches :



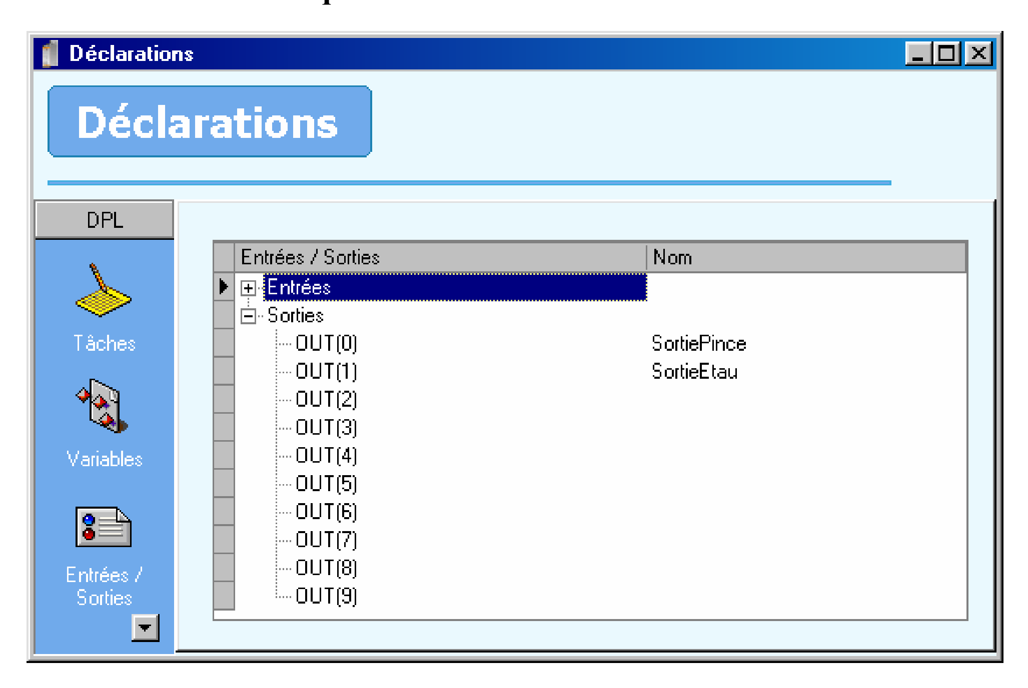
Dans cet exemple, le projet contient 3 tâches. A la mise sous tension du variateur, la tâche n°1 va démarrer de façon automatique.

Les variables :



Permet de donner un nom aux différentes variables et de l'utiliser dans les tâches DPL.

Les E/S numériques :

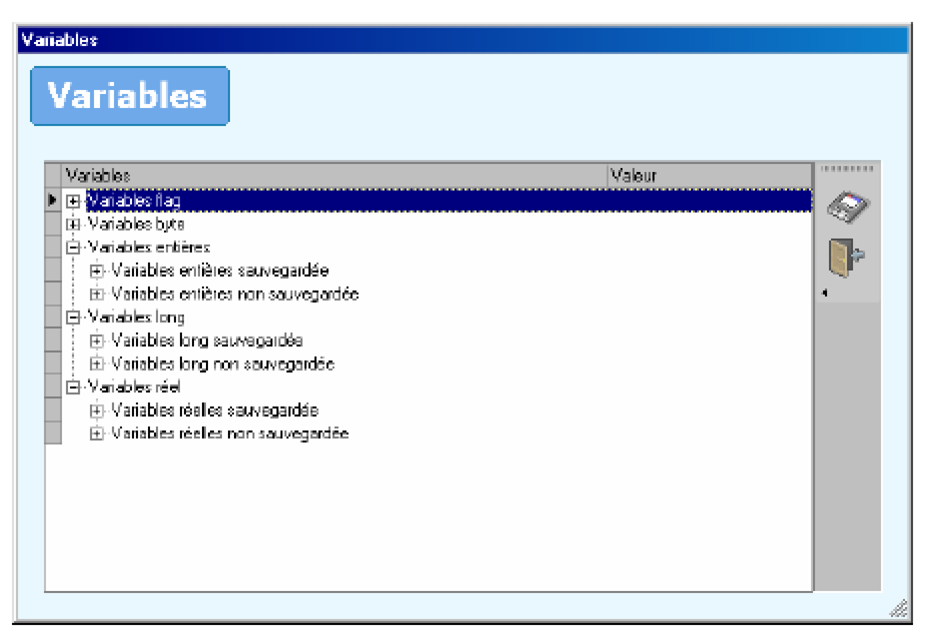


Permet de donner un nom aux différentes E/S et de l'utiliser dans les tâches DPL.

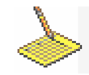
- **Editer les variables :**

Icône : 

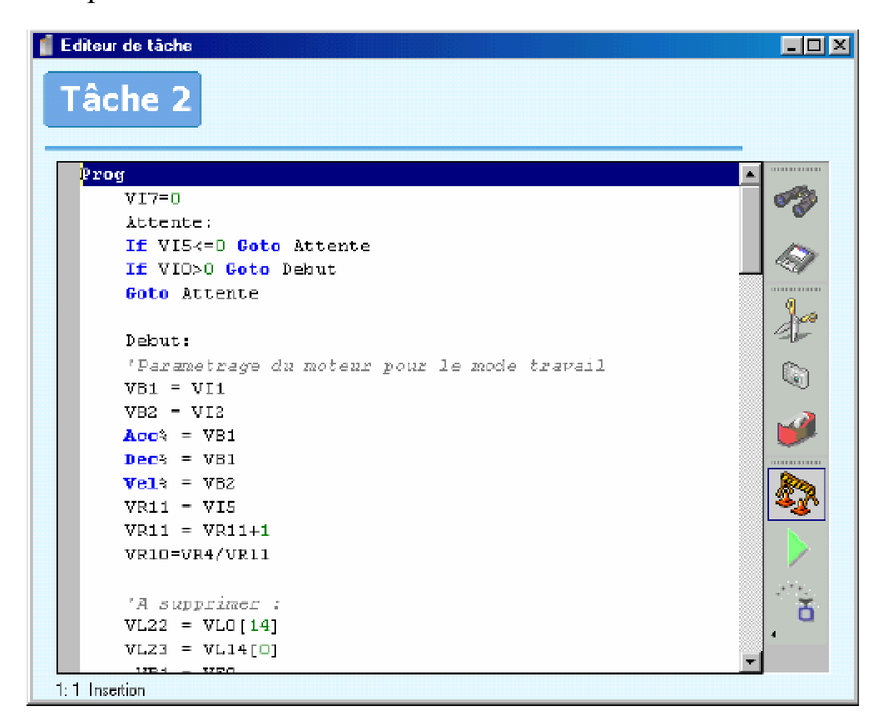
Action : Permet de visualiser et modifier les variables (contenu dans le fichier dpv du projet) et de les envoyer dans le variateur par la commande *Communication / Variables DPL / Envoyer les variables.*



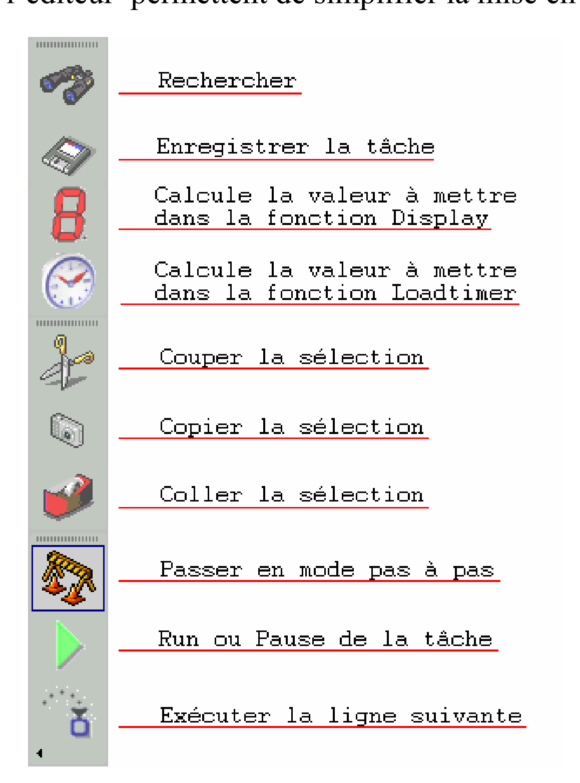
- **Editer une tâche :**

Icône : 

Action : L’éditeur de tâche se décompose en une zone d’édition de texte dans laquelle l’utilisateur vient entrer le code basic associé à son programme, une barre d’outils permettant l’aide à l’édition du code



Les outils de l’éditeur permettent de simplifier la mise en :



- **Compiler les tâches :**

Icône :

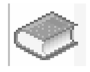
Action : Permet de compiler les tâches

- **Rechercher dans les tâches :**

Icône :

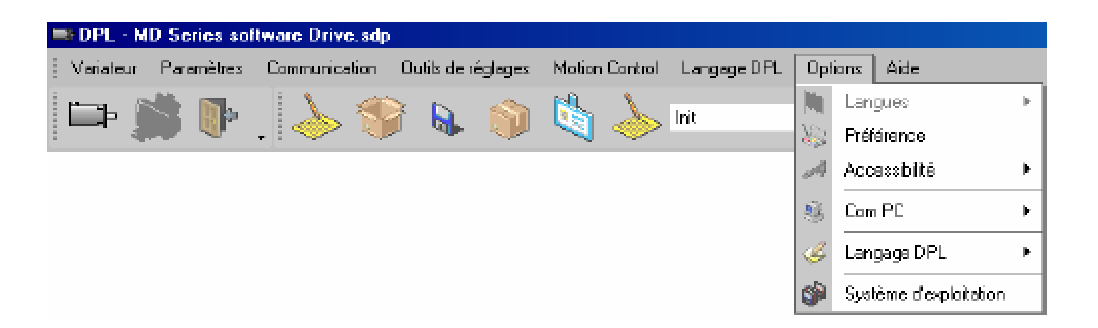
Action : Permet de rechercher une chaîne de caractères dans les tâches.

- **Informations :**

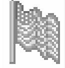
Icône : 

Action : Cette commande permet d'avoir des informations sur la mémoire programme et de rentrer des commentaires, liés au projet.

3-4-7- Options

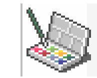


- **Langues :**

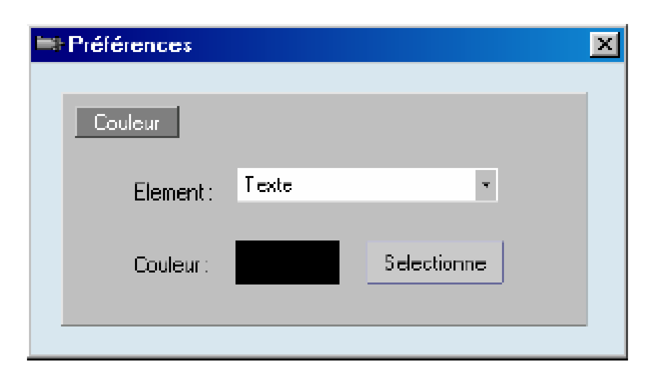
Icône : 

Action : Ce sous-menu permet de choisir la langue dans laquelle le logiciel DPL sera exploité.

- **Préférence :**

Icône : 

Action : Ce sous-menu permet de personnaliser la couleur et le fond des textes, mots clés… de l'éditeur de tâches.



La procédure de modification est la suivante : sélectionner l'un des types de texte, modifier la couleur du texte par un click sur le bouton de gauche de la souris et la couleur du fond par un click sur le bouton de droite de la souris. Un écran de visualisation permet d'observer les modifications.

- **Accessibilité :**

Icône : 

Action : Autorise l'accès aux différents niveaux de paramètres :

  - Paramètres standards
  - Paramètres avancés
  - Paramètres usine

Et permet de cacher ou rendre visible le menu DPL.

⚠ **La modification des paramètres avancés peut entraîner la détérioration du variateur. Son accès est réservé à un personnel qualifié.**

- **Com PC :**

Icône : 

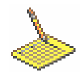
Action : Sélection du port de communication du PC : com1, com2, com3 ou com4.

L'option *Communication système* permet de forcer la communication du PC et du variateur à un format figé : 57600 bauds, 8 bits de data, 1 bit de stop, pas de parité, adresse esclave = 1.

En *communication système*, les paramètres saisis dans le menu Paramètres / Liaison RS232 de base, ne sont pas gérés.

En activant *Communication système*, le PC utilise le signal RTS et le force au niveau logique 1. Dès que le variateur lit ce signal sur son entrée CTS, son format de liaison est forcé.

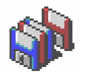
- **Langage DPL :**

Icône : 

Action : Accès aux options du langage de programmation DPL.

- Précision : définit le nombre de chiffres après la virgule pour tout ce qui est du type réel : les variables (VR0 à VR63), la position (POS_S dans le DPL) …
- Temps de vieillissement : définit le temps maximal passé dans une tâche avant de basculer vers la suivante. Il est nécessaire de recompiler les tâches après une modification.

- **Système d'exploitation :**

Icône : 

Action : Chargement d'une nouvelle version d'operating system (firmware).

Réservé à un personnel qualifié : le changement de firmware efface les paramètres du variateur. Il est ensuite nécessaire de recharger le fichier de paramètres dans le variateur.

3-4-8- Aide



- **Rubrique d'aide :**

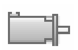
Icône : 

Action : Accès à la documentation complète.

- **Index :**

Action : Recherche rapide par famille ou mot clef.

- **A propos :**

Icône : 

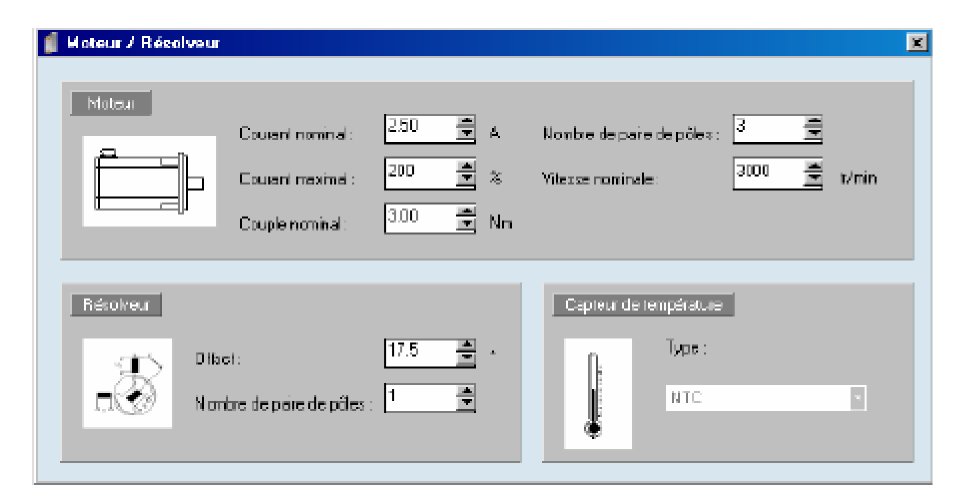
Action : Cette commande ouvre une boîte de dialogue indiquant la version du logiciel PC, la version du firmware, sa date de création, etc…

4- Réglage du variateur

4-1- Réglage des paramètres moteur et résolveur

Si vous avez transféré le fichier de paramétrage correspondant au moteur à partir de la bibliothèque, vous n'avez pas de réglage moteur/résolveur et de boucles d'asservissement à effectuer.

- Sinon sélectionner le menu « **Paramètres/moteur résolveur** ». Le menu suivant s'affiche :



4-2- Réglage moteur :

Se référer aux données du constructeur ou à la plaque signalétique du moteur.

- Entrer les valeurs du moteur (courant nominal, vitesse maximal …).

  Pour un usage normal, on mettra un courant maximal égal à 200% du courant nominal.

4-3- Réglage résolveur :

Le résolveur doit être du type TAMAGAWA TS2620N21E11 ou compatible. Pour tout autre type de résolveur, contacter notre service technique.

- Vérifier grâce à l'oscilloscope que les signaux SINUS et COSINUS de votre résolveur évolue entre +0.9 et –0.9 :

  1. Alimenter le variateur en 24V seulement (connecteur X6), le résolveur étant raccordé ainsi que la liaison RS 232.

2. Ouvrer **l'oscilloscope** dans **outils de réglage**.

3. Sélectionner les signaux SINUS et COSINUS dans RESOLVEUR puis lancer l'acquisition

4. Faites tourner le moteur à la main et visualiser les courbes obtenues. Si les signaux dépassent +0.9 ou –0.9, aller dans la liste des paramètres résolveur (accessibilité \ paramètres avancés) et baisser la valeur de *Gain excitation*. Si les signaux sont très faibles (entre +0.5 et –0.5), contacter notre service technique.

- Réglage de l'offset résolveur :

  1. Alimenter également la puissance sur le variateur.

  2. Aller dans **options** puis **accessibilité** et valider **paramètres avancés**.

  3. Aller dans **outils de réglage** et cliquer sur **auto tuning rés**olveur.

  Le variateur asservit le moteur et mesure automatiquement l'offset résolveur, cette étape ne dure que quelques secondes.

- Fermer la fenêtre de paramétrage.

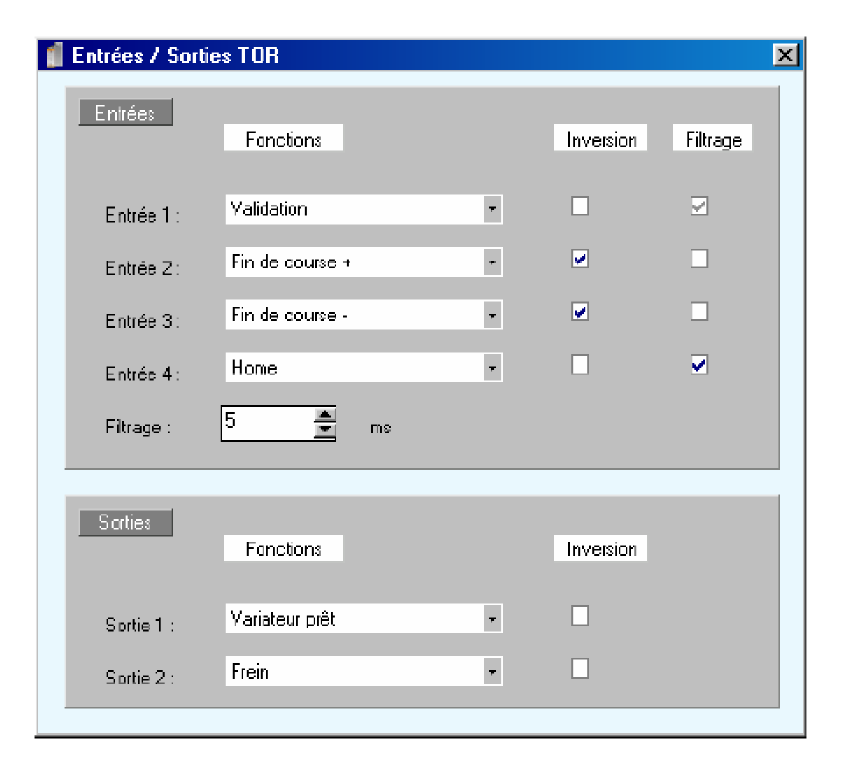
- Sauvegarder les paramètres.

Nota : Le nombre de paire de pôles du résolveur est figé à 1.

4-4- Réglage du mode de déverrouillage variateur

Pour déverrouiller le variateur, on doit sélectionner l'entrée de verrouillage. Celle-ci décide quelles conditions sont requises.

- Sélectionner le menu **Paramètres/Entrées sorties TOR.**



- Sélectionner **Aucune** dans le champ **Entrée E1**. (A la fin des réglages des boucles de régulation, penser à remodifier la fonction de l'entrée E1 selon vos besoins).

  Le bouton Enable de l'écran principal permet alors de déverrouiller ou non le variateur.

- Sauvegarder les paramètres.

4-5- Réglage des modes de fonctionnement

4-5-1- Les modes de fonctionnement

Le variateur MD gère 3 modes de fonctionnements utilisant différentes boucles de régulation.

- **MODE COUPLE** Boucle de courant

  En mode couple, le moteur maintient le couple spécifié. La vitesse dépend de la charge appliquée.

- **MODE VITESSE** Boucle de courant

  Boucle de vitesse

  En mode vitesse, le moteur maintient la vitesse spécifiée quelle que soit la charge.

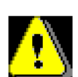
- **MODE POSITION** Boucle de courant

  Boucle de vitesse

Boucle de position

En mode position, le moteur suit un profil de trajectoire demandée.

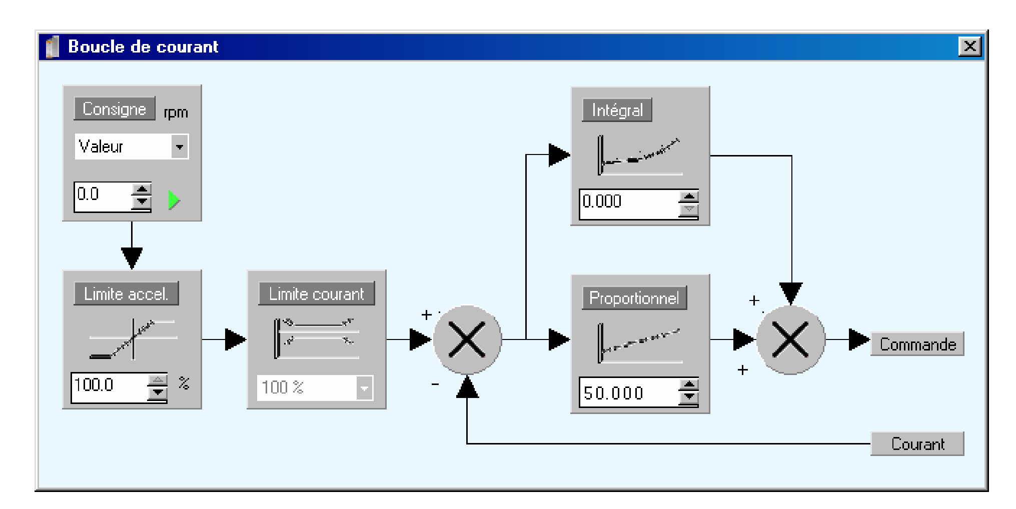
Le choix du mode de fonctionnement se fait à partir de la fenêtre PARAMETRES à la ligne variateur. Sélectionner l'un des trois modes (COUPLE, VITESSE, POSITION)

 **Le variateur doit être verrouillé lors d'un changement de mode.**

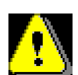
4-5-2- Réglage de la boucle de courant

Le bon réglage de la boucle de courant est indispensable pour adapter la boucle de vitesse lors des étapes suivantes. Les paramètres sont le **gain intégral** et le **gain proportionnel**. Ce réglage est directement lié aux caractéristiques du moteur et ne dépend pas de la charge.

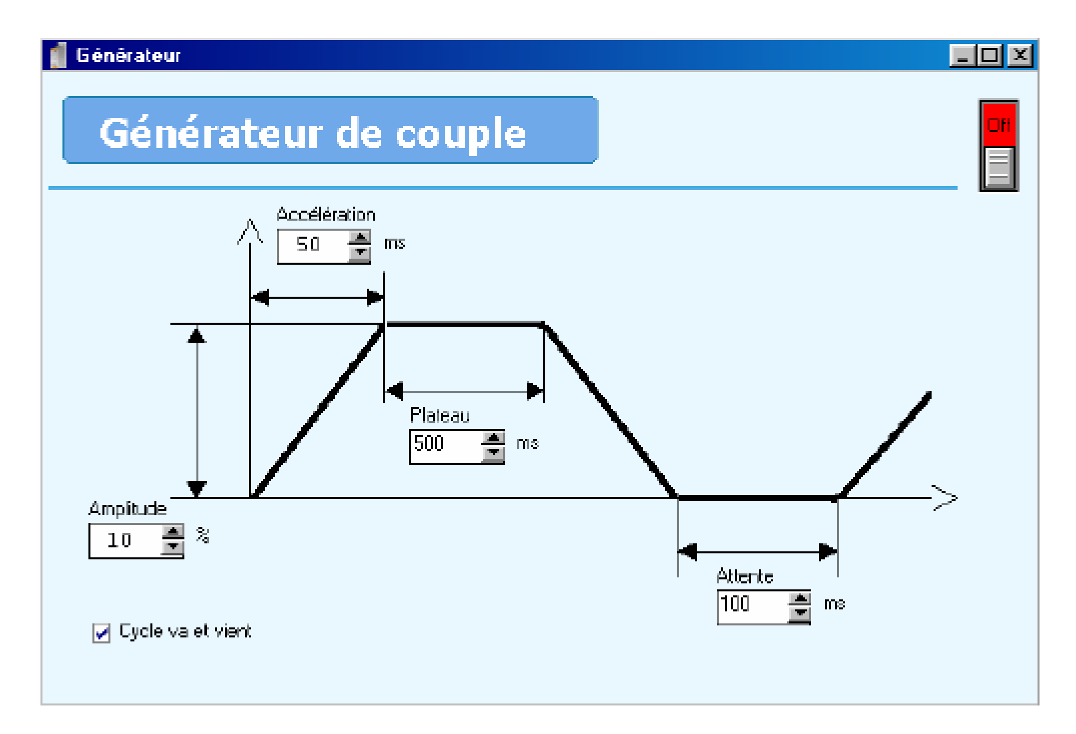
- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode couple à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres / Boucle de courant**. Le menu suivant s'affiche :



Pour commencer le réglage de la boucle de courant, prendre les réglages ci dessus.

 La consigne doit être du type **valeur**.

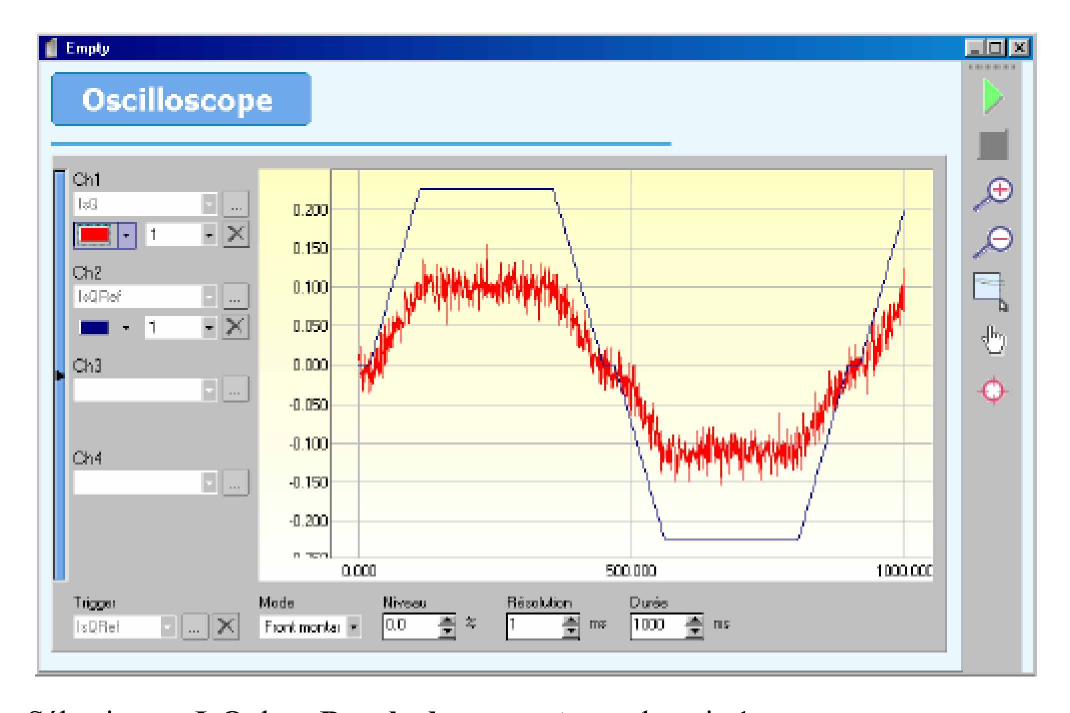
- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :



Vous pouvez faire varier l'amplitude de 5 à 15 % et l'accélération de 50 à 100%, selon le type de moteur. L'amplitude est exprimée en pourcentage du courant maximal du moteur.

⚠ Pour lancer un mouvement vous devez asservir le variateur par le bouton *Enable* en position ON sur l'écran principal.

- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe du courant durant le mouvement :



1. Sélectionner **IsQ** dans **Boucle de courant** pour la voie 1.
2. Sélectionner **IsQREF** dans **Boucle de courant** pour la voie 2.
3. Sélectionner **IsQREF** pour le trigger et choisir front montant.

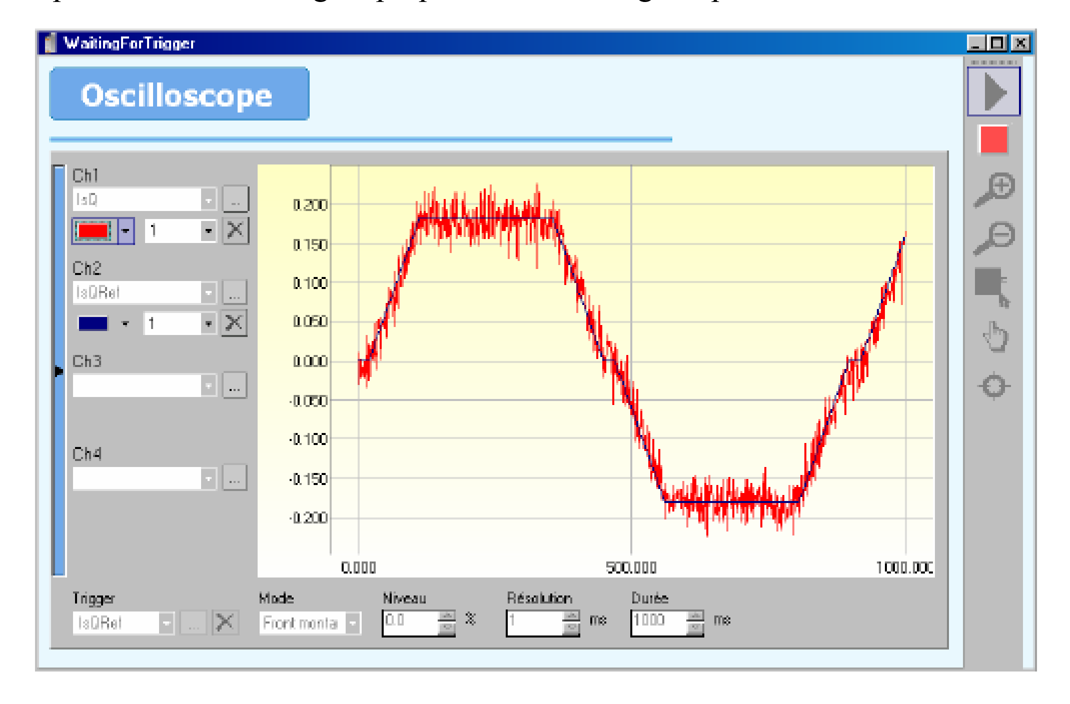
Si le signal IsQREF n'a pas la forme d'un trapèze modifier alors les valeurs *Amplitude* et *Accélération* dans la fenêtre oscilloscope.

- Avant de commencer, il est préférable de bloquer l'arbre du moteur (par exemple avec la main dans le cas de petits moteurs).

1. Augmenter le gain proportionnel jusqu'à ce que le courant réel (IsQ) s'approche le plus près possible de la consigne (IsQREF).
2. Si le moteur se met à vibrer, baisser le gain proportionnel de 20%.
3. Augmenter le gain intégral jusqu'à ce que le courant réel suive parfaitement la consigne.


Valeurs usuelles : gain proportionnel de 30 à 500, gain intégral de 1 à 10.

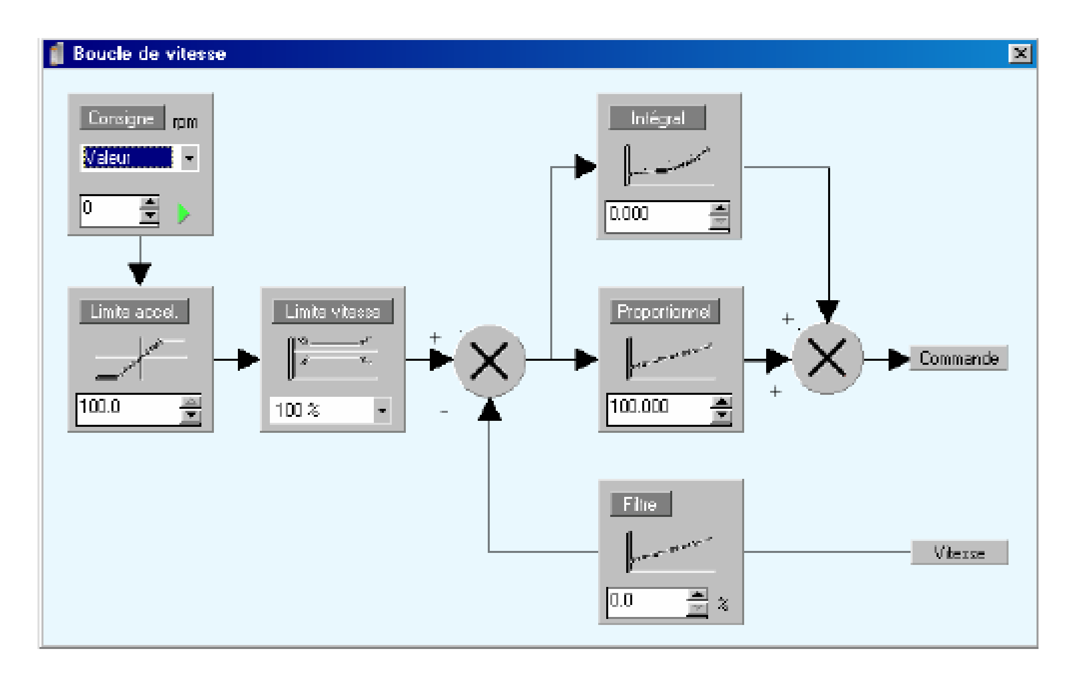
Exemple de courbes avec gain proportionnel et intégral optimisés :



- Sauver les réglages avec **Paramètres/Sauvegarder les paramètres**.

4-5-3- Réglage de la boucle de vitesse

- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode **vitesse** à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres / Boucle de vitesse**

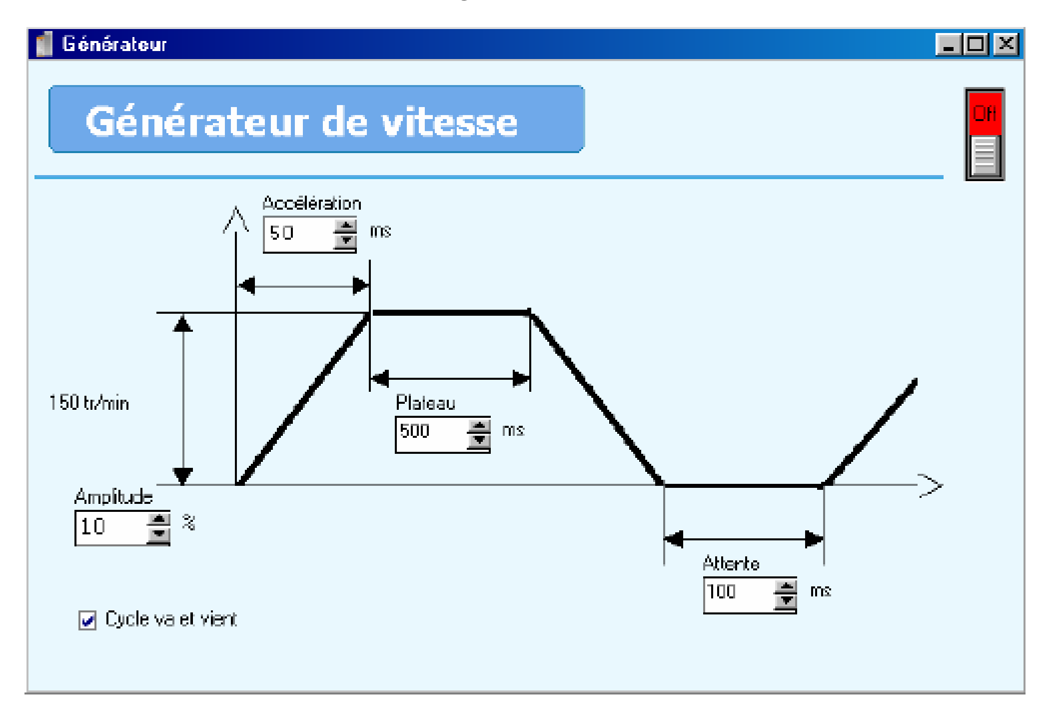


Pour commencer le réglage de la boucle de vitesse, prendre les réglages ci dessus.

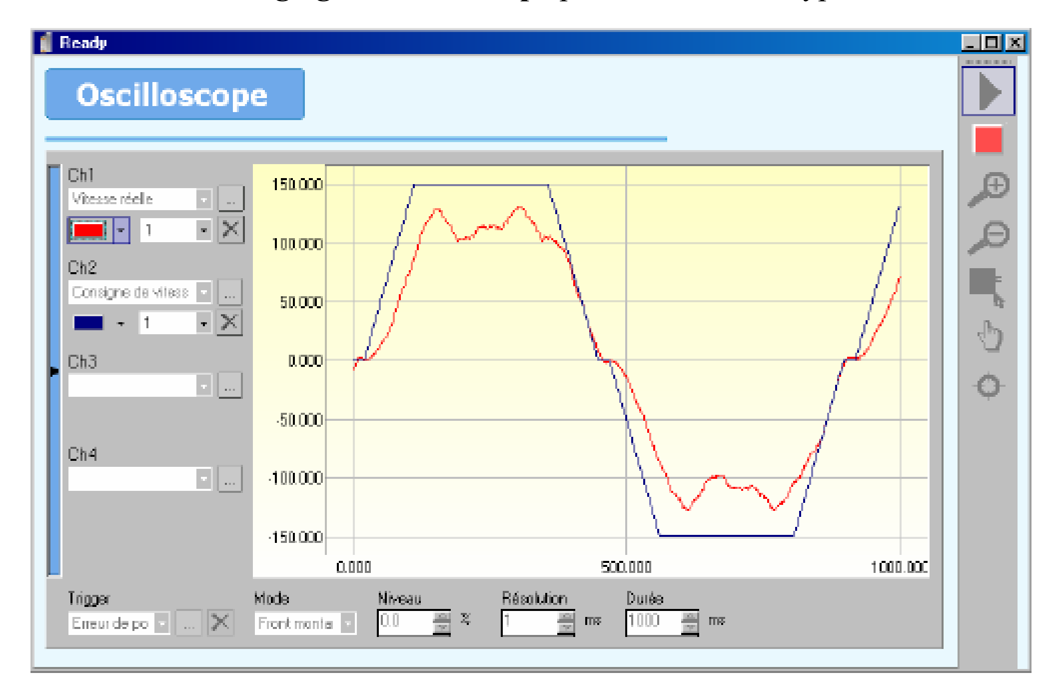
 La consigne doit être du type **valeur**.

- Déverrouiller le variateur (bouton *Enable* sur ON dans l'écran principal).
- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :

L'arbre du moteur ne doit pas être bloqué. Un réglage optimal de la boucle de vitesse, s'effectue avec le moteur charge.



- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe de vitesse :



1. Sélectionner **Vitesse réelle** dans **Boucle de vitesse** pour la voie 1.
2. Sélectionner **Consigne de vitesse** dans **Boucle de vitesse** pour la voie 2.
3. Sélectionner **Consigne de vitesse** pour le trigger et choisir front montant.

Si le signal Consigne de vitesse n'a pas la forme d'un trapèze, modifier alors les valeurs d'**Amplitude** et d'**Accélération** dans la fenêtre générateur.

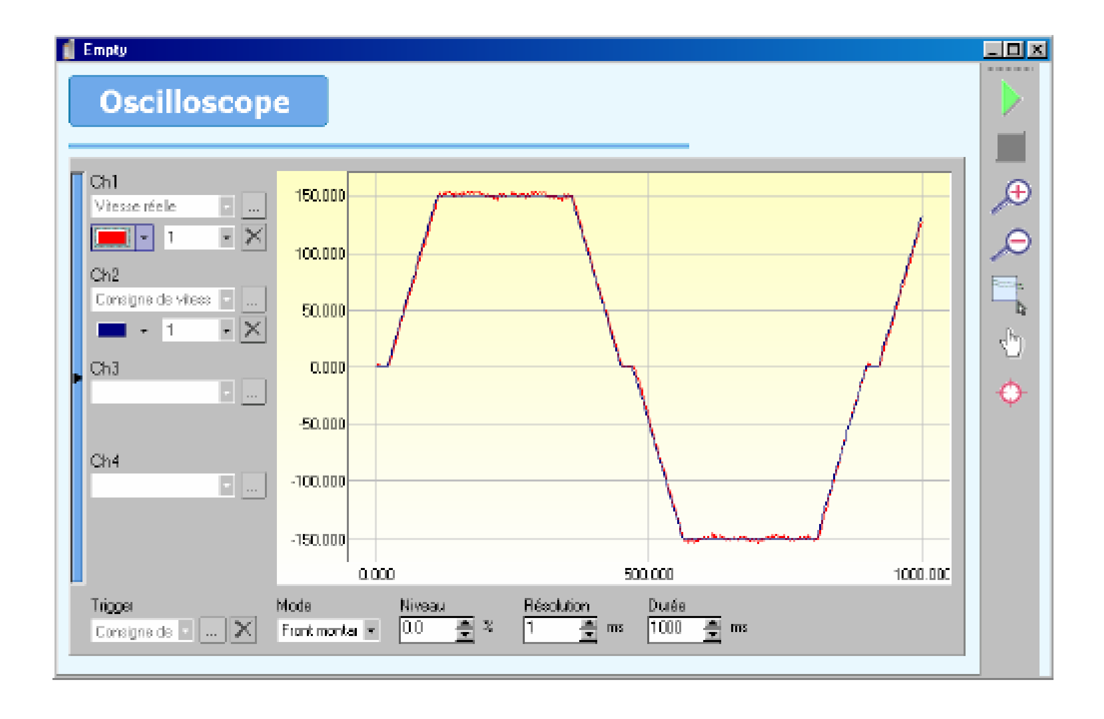
- Augmenter le **gain proportionnel** jusqu'à ce que la vitesse réelle s'approche le plus près possible de la consigne.

Si le moteur se met à vibrer, baisser le **gain proportionnel** de 20%.

Augmenter le **gain intégral** jusqu'à ce que la vitesse réelle suive parfaitement la consigne.

Valeurs usuelles : gain proportionnel de 200 à 1000, gain intégral de 1 à 20.

Exemple de courbes avec gain proportionnel et intégral optimisés :

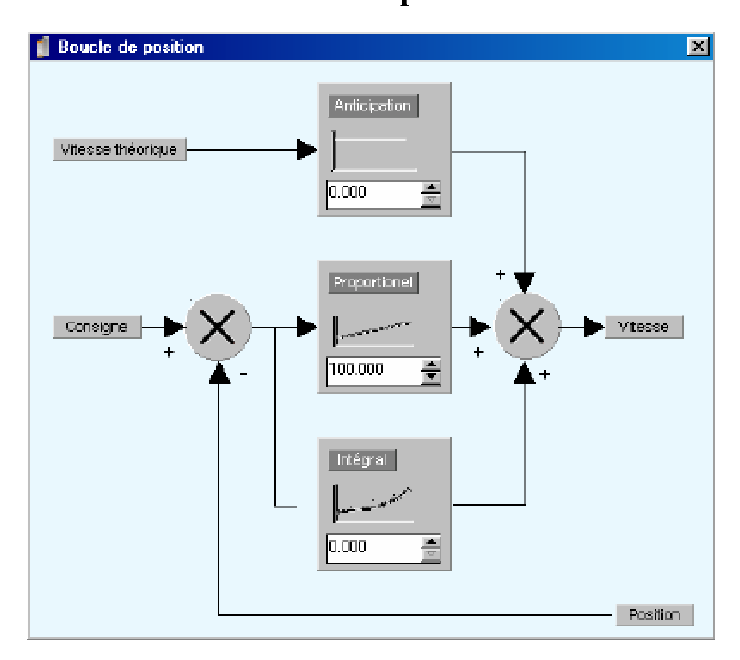


- Sauver les paramètres avec **Paramètres/Sauvegarder les paramètres.**

4-5-4- Réglage de la boucle de position

Le réglage de la boucle de position se fait en demandant des déplacements à partir de la fenêtre Générateur.

- Verrouiller le variateur (bouton *Enable* sur OFF dans l'écran principal).
- Sélectionner le variateur en mode **Position** à partir de la fenêtre principale.
- Sélectionner le menu **Paramètres/Boucle de position**

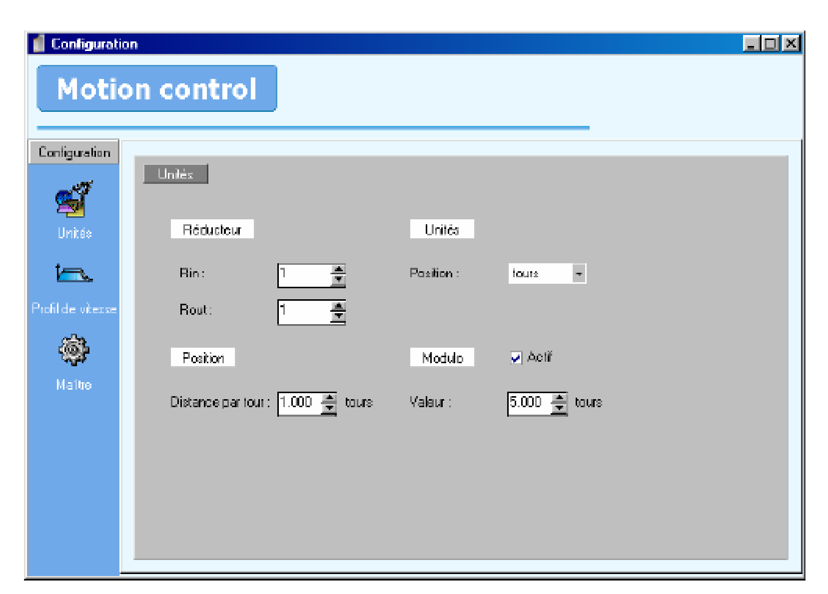


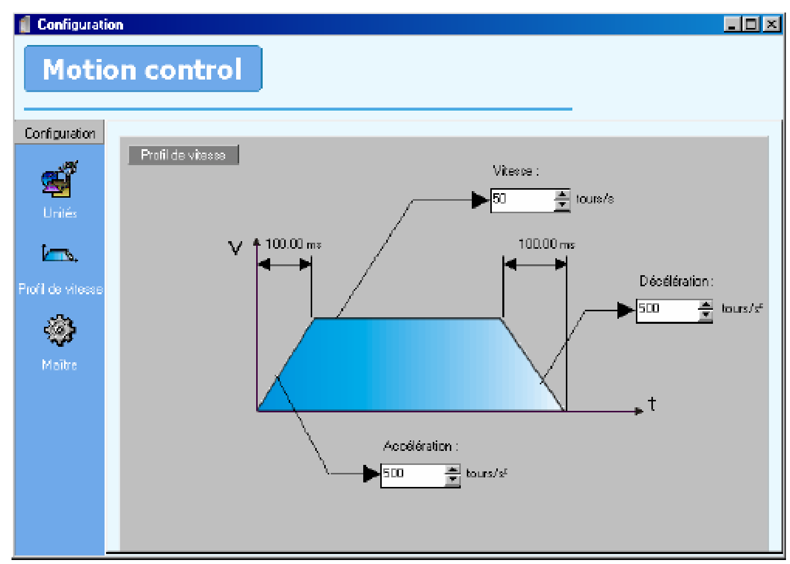
Pour commencer le réglage de la boucle de position, prendre les réglages ci dessus.

- Dans **Motion control** / **Configuration**, modifier les unités et le profil de vitesse pour correspondre à votre besoin :

Exemple pour un moteur avec une vitesse nominal de 3000 tr/min

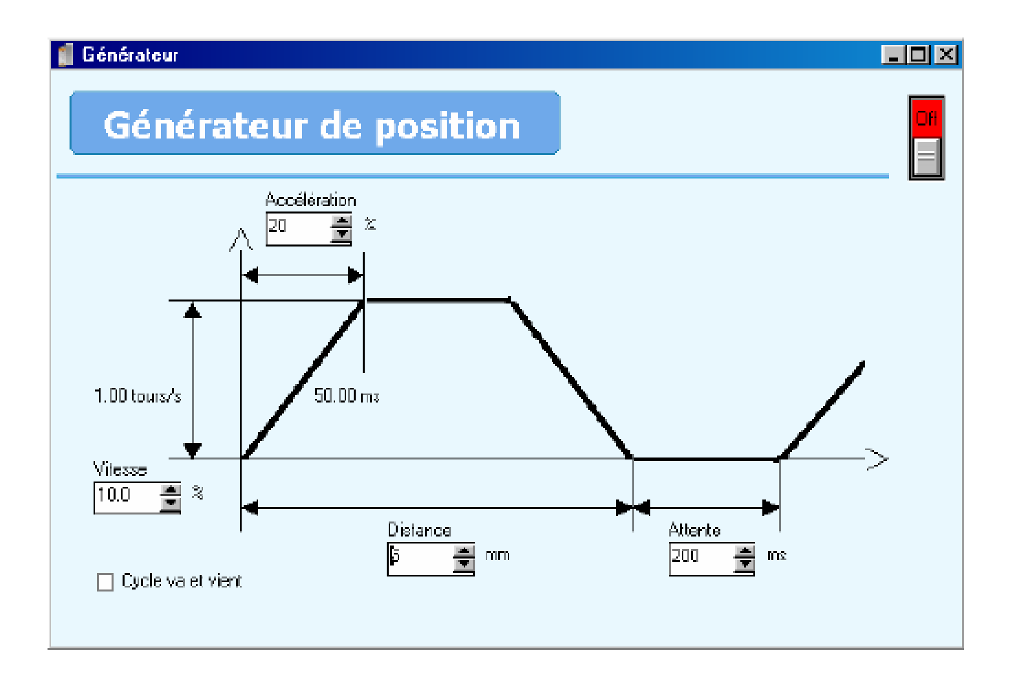
Le pourcentage de vitesse et d'accélération que l'on rentre dans la fenêtre du générateur fait référence à la vitesse et à l'accélération données dans le menu **Motion control** / **Configuration** / **Profil de vitesse.**



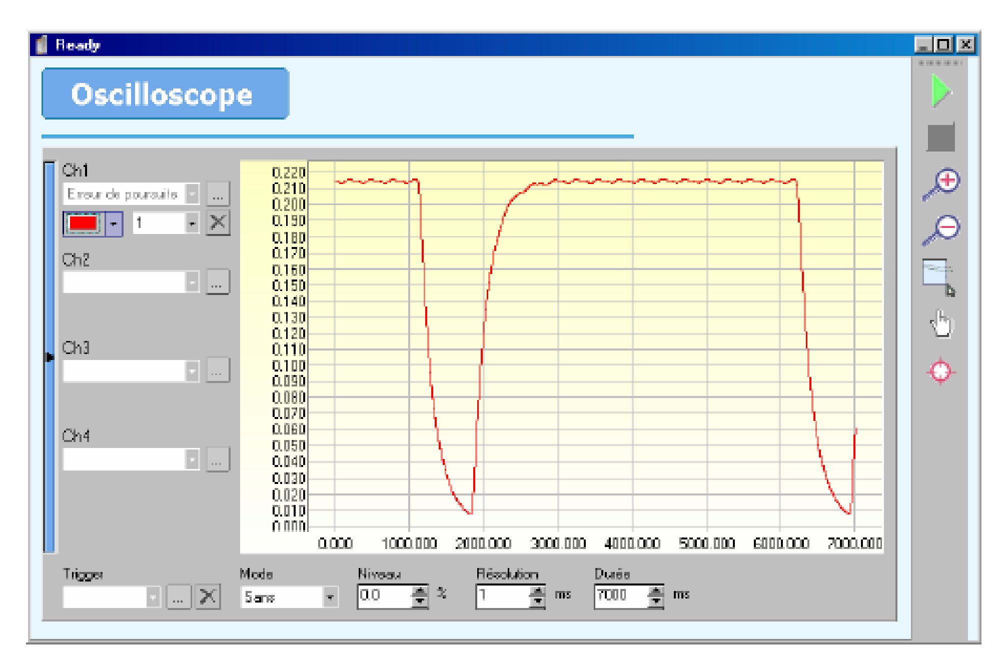


Selon les caractéristiques de votre moteur, pensez à régler votre erreur de poursuite dans **Paramètres** / **Sécurité** / **Position** / **Erreur de poursuite**

- Dans **Outils de réglages / Générateur**, lancer un mouvement comme ci dessous :



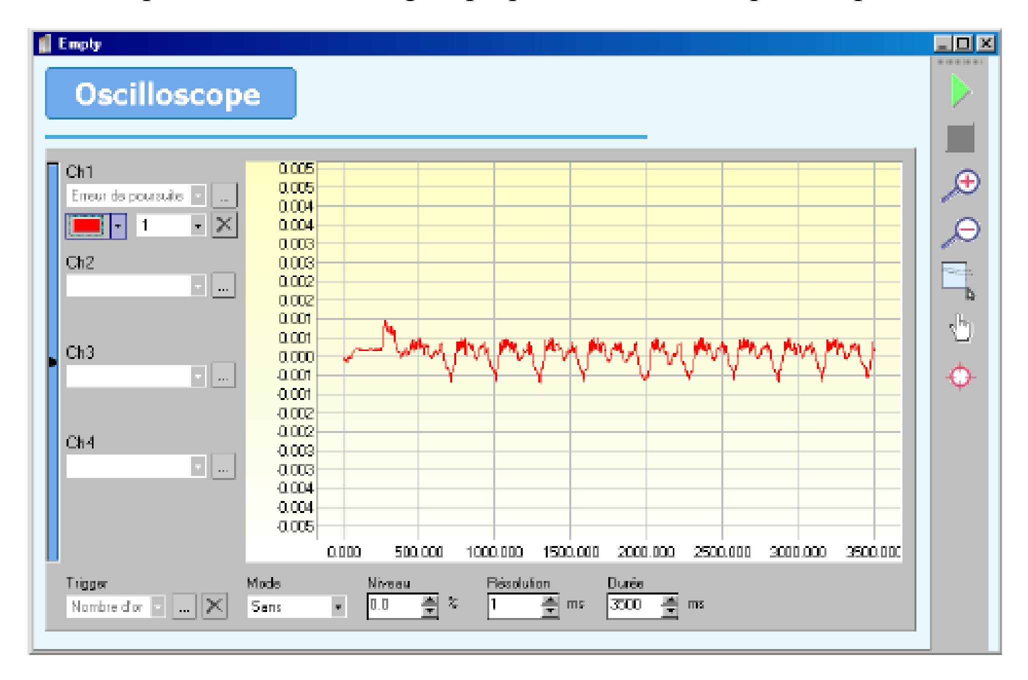
- Aller dans **Outils de réglages / Oscilloscope** pour visualiser ce type de courbe d'erreur de poursuite :



- Sélectionner **Erreur de poursuite** dans **Boucle de position** pour la voie 1.
- Ne pas sélectionner de fonction trigger.
- Augmenter le **gain proportionnel** jusqu'à atteindre la limite de la stabilité du moteur puis le baisser de 20%.
- Augmenter le **gain anticipation** de vitesse pour tendre vers une erreur de poursuite nulle.

 Valeurs usuelles : gain proportionnel de 1000 à 3000, gain anticipation de 60 à 65.

Exemple de courbes avec gain proportionnel et anticipation optimisés



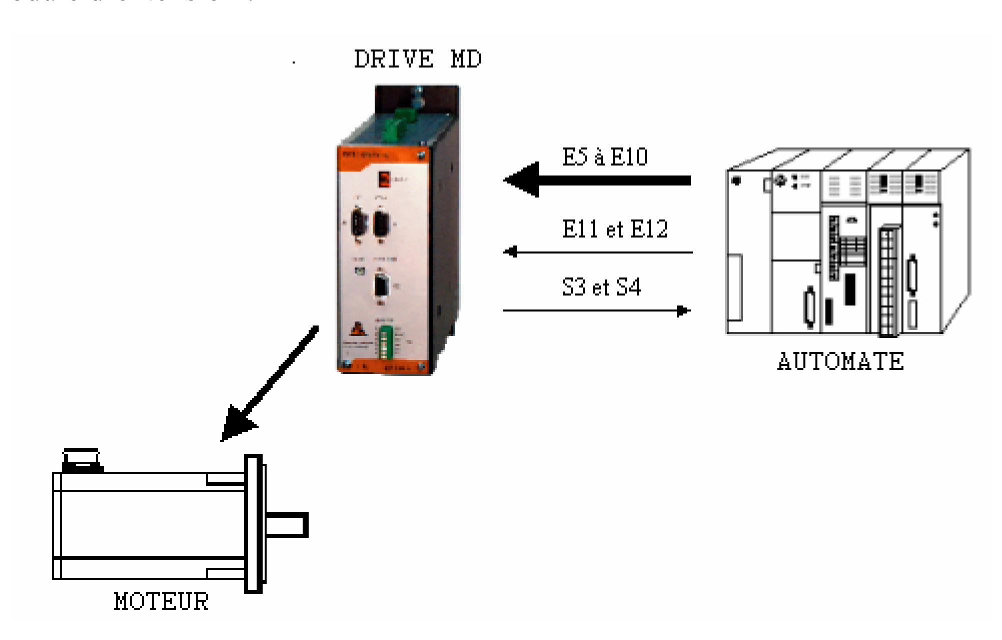
Nota : Il est également intéressant de visualiser sur le canal n°2 de l'oscilloscope la vitesse théorique afin de connaître la valeur de l'erreur de poursuite pendant les phases d'accélération et de décélération. Dans ce cas régler le canal n°1 avec un facteur de 1000 et le canal n°2 avec un facteur de 0.001

- Sauver les paramètres avec **Paramètres/Sauvegarder les paramètres.**

5- Les trajectoires

5-1- Introduction :

Le mode trajectoire permet à un automate ou un boîtier de commande externe de lancer des mouvements (jusqu'à 64, préenregistrés dans une table) à partir des entrées logiques du module d'extension :



Pour chaque profil de trajectoire, on peut définir une vitesse, une accélération et une décélération. Tous ces paramètres sont stockés dans les 64 premières variables de type réel et entier long.

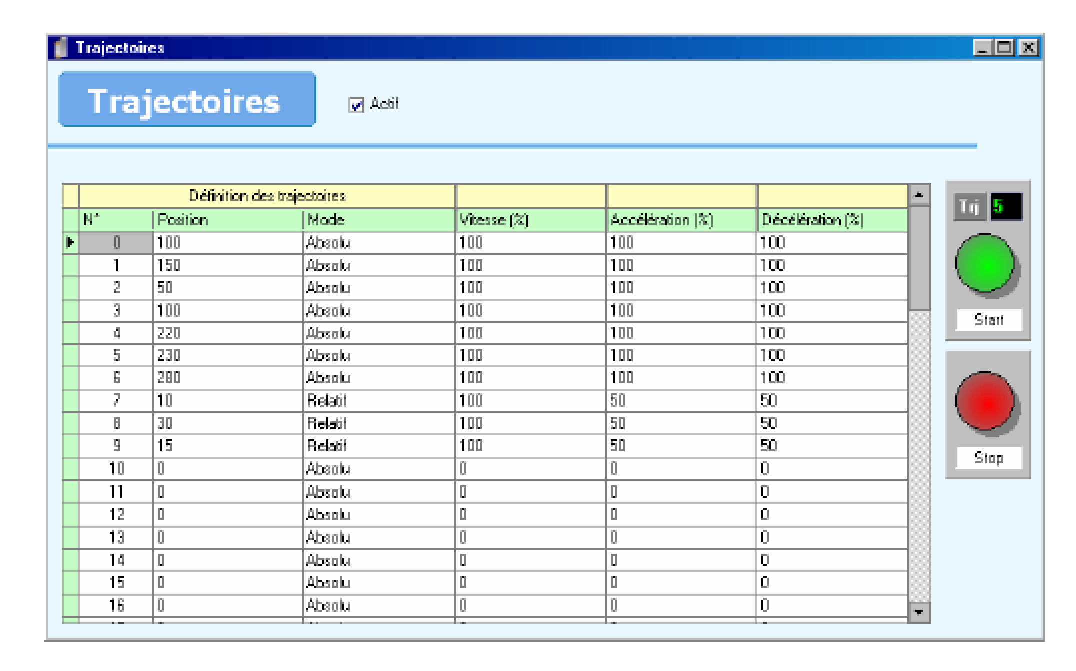
Si vous utilisez le DPL en même temps que les trajectoires, la modification des variables : VR0 à VR63 ou VL0 à VL63 par les tâches DPL modifiera aussi les trajectoires correspondantes.

5-2- Mise en oeuvre :

- **Définition des trajectoires :**

Pour avoir accès aux trajectoires, il faut que le variateur soit en mode position.

- Cliquer sur **Trajectoires** dans le menu **Motion Control**.
- Si le variateur est connecté au PC, ce dernier va chercher les trajectoires contenues dans le variateur et les affiches sinon il vous demande d'ouvrir un fichier de trajectoires ou d'en créer un.



Définition des trajectoires					
N°	Position	Mode	Vitesse (%)	Accélération (%)	Décélération (%)
0	100	Absolu	100	100	100
1	150	Absolu	100	100	100
2	50	Absolu	100	100	100
3	100	Absolu	100	100	100
4	220	Absolu	100	100	100
5	230	Absolu	100	100	100
6	280	Absolu	100	100	100
7	10	Relatif	100	50	50
8	30	Relatif	100	50	50
9	15	Relatif	100	50	50
10	0	Absolu	0	0	0
11	0	Absolu	0	0	0
12	0	Absolu	0	0	0
13	0	Absolu	0	0	0
14	0	Absolu	0	0	0
15	0	Absolu	0	0	0
16	0	Absolu	0	0	0

- Sélectionner le mode d'utilisation des trajectoires (désactivé, normal ou avancé).
- Pour chaque trajectoire vous devez entrer :

  1. une position
  2. un mode : absolu, relation, infini +, infini – ou home
  3. une vitesse en %
  4. une accélération en %
  5. une décélération en %

⚠ Toutes les valeurs saisies dépendent des **unités** et **profil de vitesse** entrés dans **Motion Control / Configuration.**

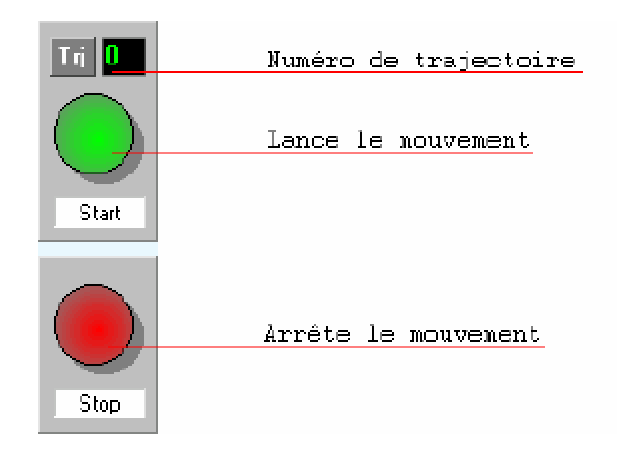
Pour exécuter une prise d'origine à partir des trajectoires :

1. Déclarer une trajectoire en mode **HOME.**
2. Paramétrer la prise d'origine dans **Motion Control / Home.**
3. Paramétrer l'entrée E4 en fonction **Home** dans **Paramètres \ E/S Logiques**, si vous utilisez un capteur de prise d'origine.

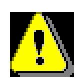
Sauver les trajectoires avec **Communication** / **Trajectoires** / **Sauver les trajectoires**.

- **Simulation des trajectoires :**

Dans l'écran **Définition des trajectoires,** vous pouvez simuler les trajectoires saisies :



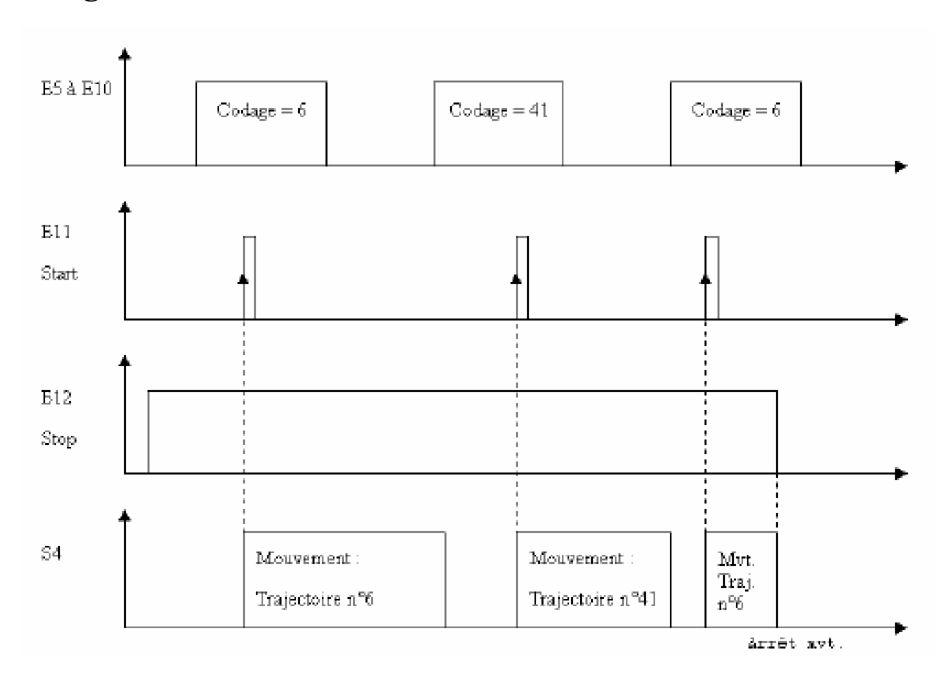
1. Vérifier que le variateur est asservi et qu'un des modes trajectoire soit actif.
2. Cliquer sur le numéro de la trajectoire à exécuter.
3. Appuyer sur START pour lancer la trajectoire.
4. Appuyer sur STOP si l'on souhaite arrêter le mouvement avant la fin.

 L'entrée STOP (E12) doit être à niveau logique 1 pour autoriser une trajectoire.

- **Les fichiers TRJ :**
  - Il est possible d'enregistrer les trajectoires contenues dans le variateur vers un fichier .trj avec **Communication** / **Trajectoires** / **Recevoir les trajectoires.**
  - De la même manière, il est possible de transférer les trajectoires contenues dans un fichier .trj vers le variateur avec **Communication** / **Trajectoires** / **Envoyer les trajectoires.**

5-3- Trajectoire en mode normal :

5-3-1- Chronogrammes :



5-3-2- Carte d'extension I/O :

- De E5 à E10 : 6 entrées pour le codage du numéro de trajectoire, avec E5 étant le bit de poids faible et E10 le bit de poids fort.
- E11 : entrée START sur front montant déclenchant le mouvement.
- E12 : entrée STOP, à niveau logique 1 en fonctionnement. Si passage à niveau logique 0, tout mouvement en cours s'arrête.
- S3 : sortie image de la prise d'origine (HOME_S) : 0 si home non fait et 1 si fait
- S4 : sortie image du mouvement (MOVE_S) : 0 si axe à l'arrêt et 1 si axe en mouvement.

Attention : E5 correspond à la 1er entrée du module d'extension I/O

5-3-3- Composition d'une trajectoire :

Chaque trajectoire est codée sur un réel et un entier long.

Ex : La trajectoire TRJ0 est codée sur VR0 et VL0

La trajectoire TRJ19 est codée sur VR19 et VL19

- La variable réelle contient la position de la trajectoire.

- L'entier long est divisé en 4 octets suivants :

  1er octet : le mode (poids fort)

  - 0 : absolu
  - 1 : relatif
  - 2 : infini +
  - 3 : infini –
  - 4 : home

  2ième octet : la vitesse (en %)

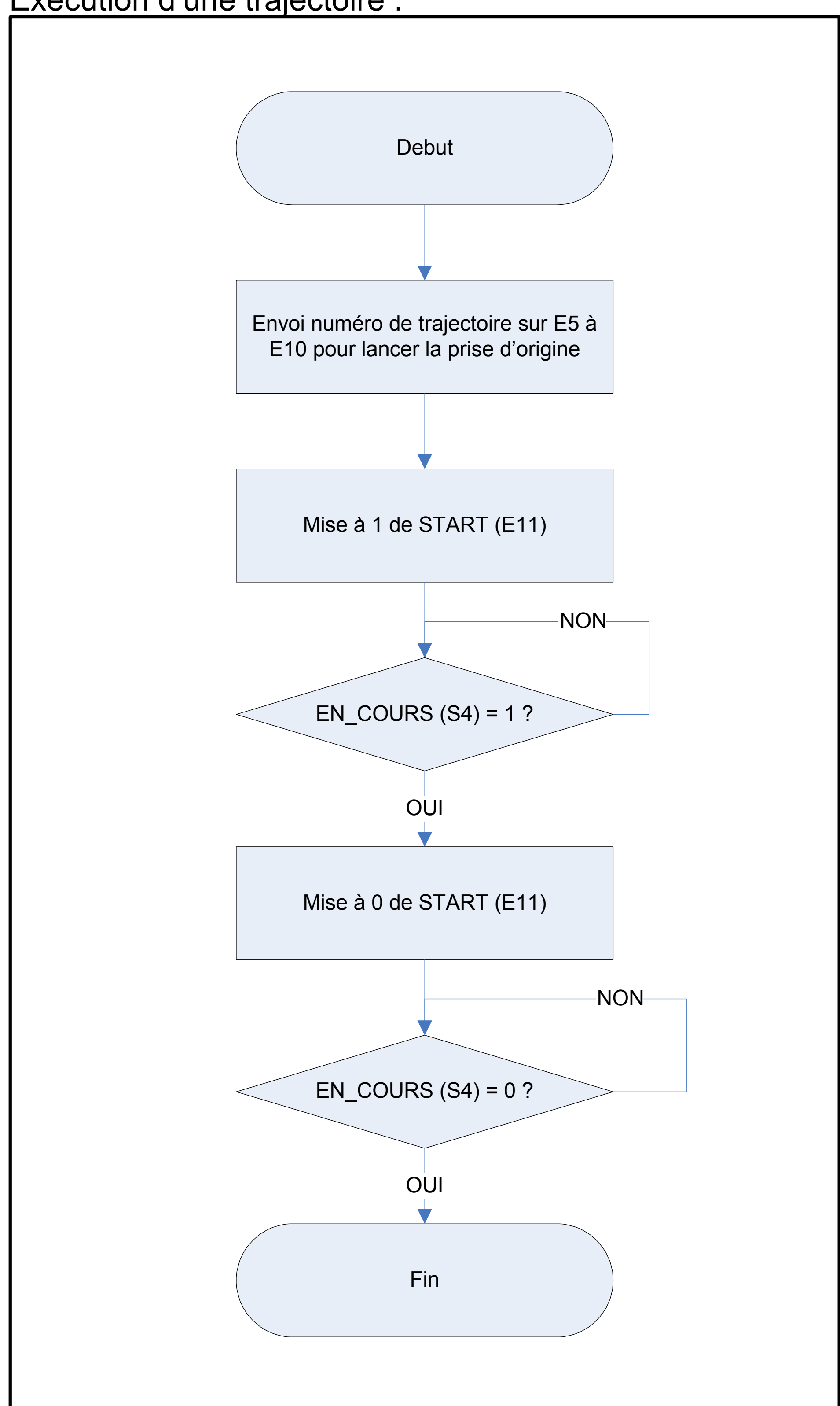
  3ième octet : l'accélération (en %)

  4ième octet : la décélération (poids faible) (en %)

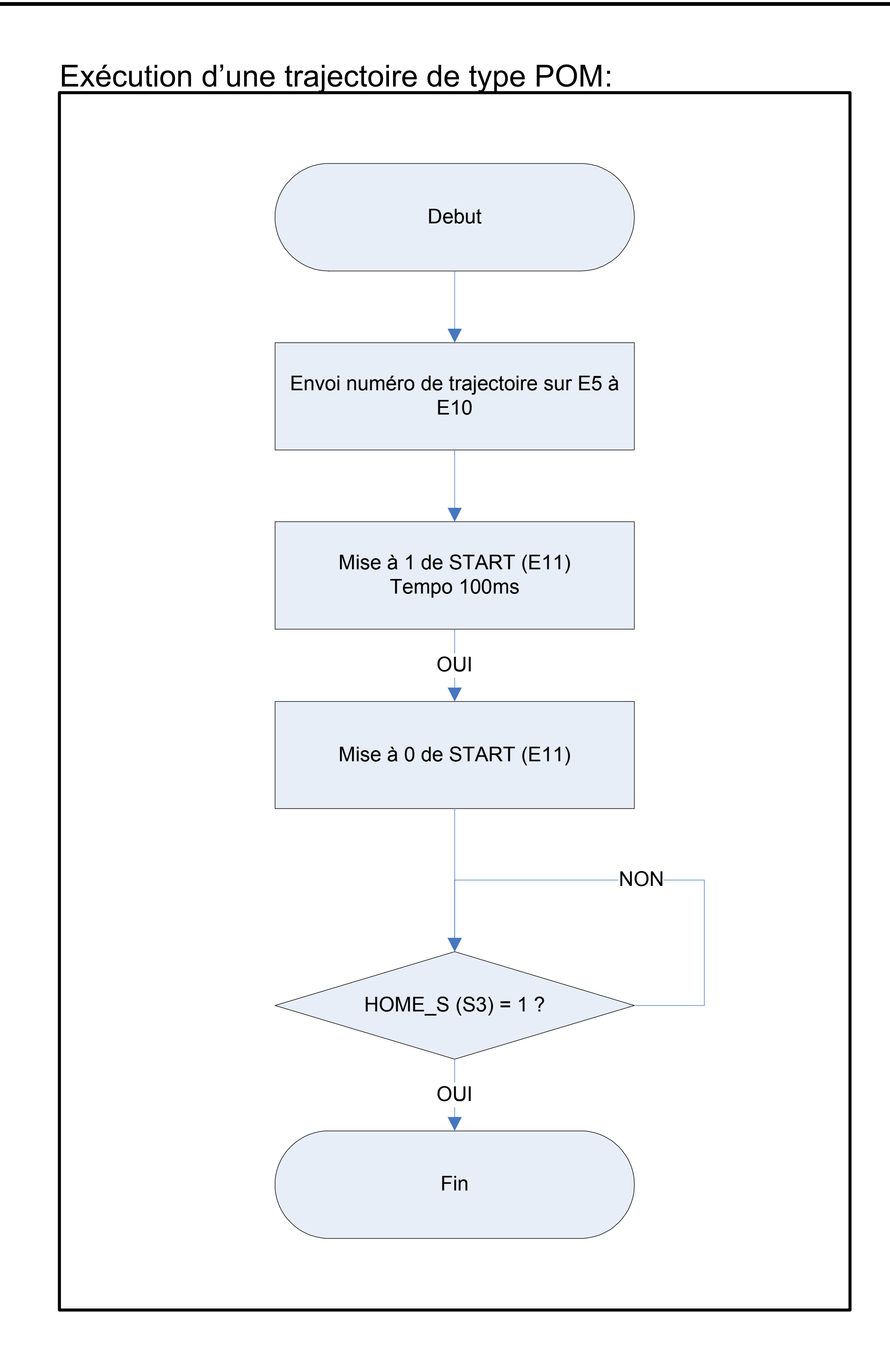
5-4- Trajectoire en mode avancé :

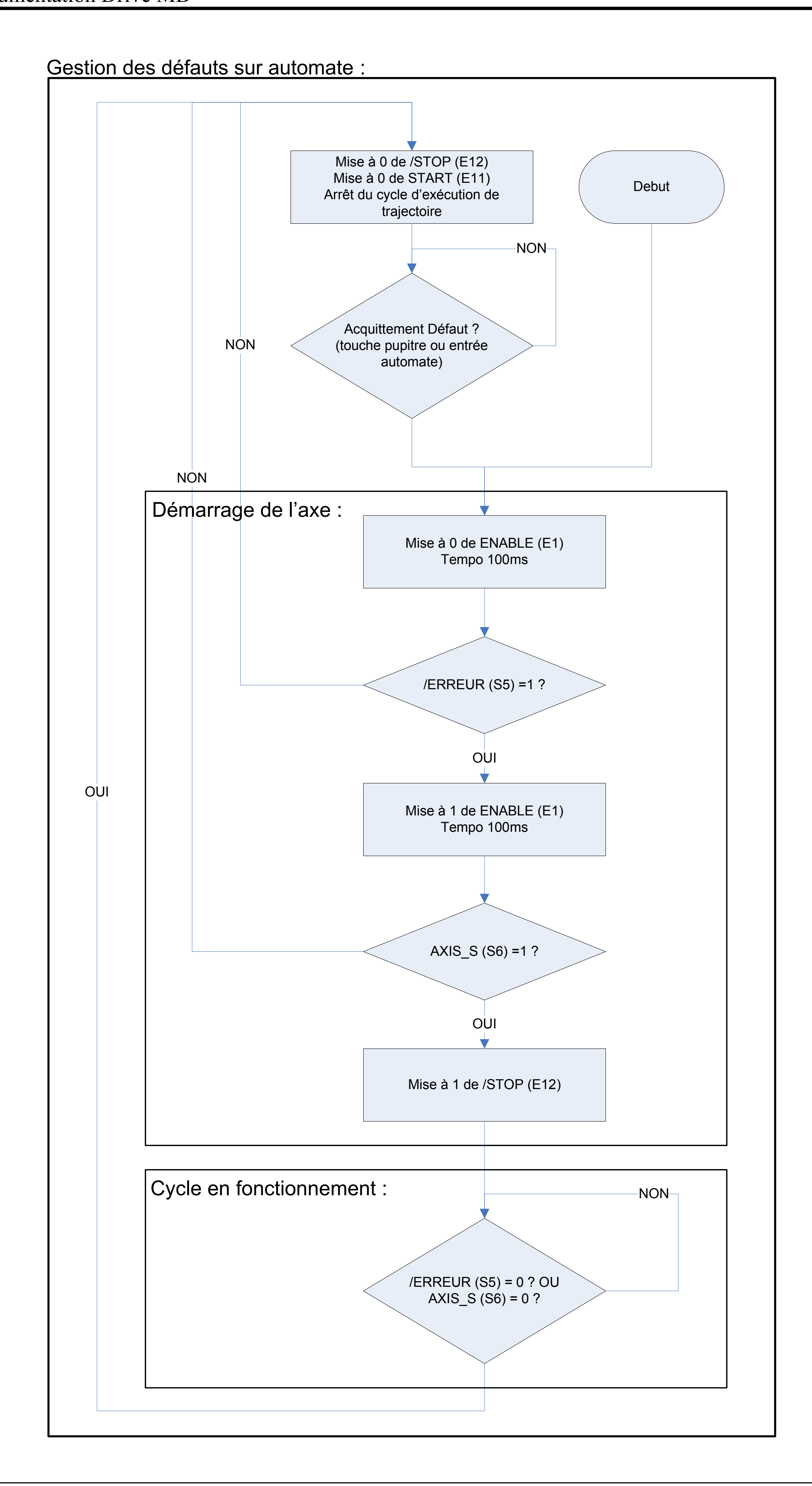
5-4-1- Organigrammes :

Exécution d'une trajectoire :



Exécution d'une trajectoire de type POM:




Gestion des défauts sur automate :
Mise à 0 de /STOP (E12)
Mise à 0 de START (E11)
Arrêt du cycle d'exécution de trajectoire
Debut
NON
Acquittement Défaut ?
(touche pupitre ou entrée automate)
NON
NON
Démarrage de l'axe :
Mise à 0 de ENABLE (E1)
Tempo 100ms
/ERREUR (S5) =1 ?
OUI
OUI
Mise à 1 de ENABLE (E1)
Tempo 100ms
AXIS_S (S6) =1 ?
OUI
Mise à 1 de /STOP (E12)
Cycle en fonctionnement :
NON
/ERREUR (S5) = 0 ? OU
AXIS_S (S6) = 0 ?

5-4-2- Entrées/sorties logiques :

De base :

- E1 : entrée ENABLE permet d'asservir le variateur sur front montant et de désasservir l'axe sur état 0 (l'entrée 1 doit être déclarée en fonction **VALIDATION** dans **Paramètres \ E/S Logiques**).

Sur extension I/O :

- De E5 à E10 : 6 entrées pour le codage du numéro de trajectoire, avec E5 étant le bit de poids faible et E10 le bit de poids fort.
- E11 : entrée START sur front montant déclenchant le mouvement.
- E12 : entrée STOP, à niveau logique 1 en fonctionnement. Si passage à niveau logique 0, tout mouvement en cours s'arrête.
- S3 : sortie image de la prise d'origine (HOME_S) : 1 si la POM est non faite sinon 0
- S4 : sortie image de la trajectoire (EN_COURS) : 0 si aucune trajectoire est en cours et 1 si l'axe en mouvement (trajectoire en cours) ou en attente d'acquittement.
- S5 : sortie image erreur de trajectoire (/ERREUR) : 0 si erreur de trajectoire et 1 si pas de défaut.
- S6 : sortie image de l'état de l'asservissement (AXIS_S).

Attention : E5 correspond à la 1er entrée du module d'extension I/O

5-4-3- Composition d'une trajectoire :

Chaque trajectoire est codée sur un réel et un entier long.

Ex : La trajectoire TRJ0 est codée sur VR0 et VL0

La trajectoire TRJ19 est codée sur VR19 et VL19

- La variable réelle contient la position de la trajectoire.
- L'entier long est divisé en 4 octets suivants :

1er octet : le mode (poids fort)

- 0 : absolu
- 1 : relatif
- 2 : infini +
- 3 : infini –

- 4 : home

2ième octet : la vitesse (en %)

3ième octet : l'accélération (en %)

4ième octet : la décélération (poids faible) (en %)

6- Langage de programmation DPL

6-1- Introduction

6-1-1- Introduction

- Le langage DPL (Drive Programming Language) est un outil de programmation puissant et simple à utiliser. Il offre une architecture structurée rencontrée sur les langages de haut niveau. Pour une programmation flexible, ce langage est géré par un noyau temps réel multitâches, utilisant des instructions pseudo-basic et contenant également toutes les fonctions de contrôle de mouvement et d'automate.

- Le langage intègre aussi la gestion de données sous la forme de variables.

- Un projet développé à partir du DPL peut contenir jusqu'à 4 tâches fonctionnant en parallèle. Chaque tâche possède un niveau de priorité et est écrite en basic.

6-1-2- Affectation du plan mémoire

Affectation de la mémoire FLASH

Zone réservée au système : ➪ Boot ➪ Système d'exploitation (Firmware)
256 paramètres système : ➪ 512 octets
Valeurs variables initialisées : => 512 octets ➪ VR0 à VR63 ➪ VL0 à VL63
Programme BASIC : ➪ 7 Ko 4 tâches motion-basic

Affectation de la mémoire RAM

Zone réservée au système : ➪ Boot ➪ Système d'exploitation
256 variables de type réel : ➪ 1Ko ➪ de VR0 à VR255
256 variables de type entier long signé : ➪ 1 Ko ➪ VL0 à VL255
256 variables de type entier non signé : ➪ 512 octets ➪ VI0 à VI255
256 variables de type octet : ➪ 256 octets ➪ VB0 à VB255
256 variables de type flag : ➪ 32 octets ➪ VF0 à VF255

6-2- Les données

6-2-1- Variables

Toute variable est globale et peut être utilisée par plusieurs tâches.

Elle peut aussi être traitée comme un tableau (notion d'indexage).

On peut attribuer un nom à une variable à partir de l'écran Langage DPL / Déclarations / Variables et l'utiliser dans les tâches DPL.

Ex : Position = POS_S

Les variables sont numérotées de 0 à 255.

Tableau récapitulatif des différents types :

Type	Valeur	Occupation mémoire	Exemple
Flag (Bit)	1/0, On/Off ou True/False	1 bit à l'intérieur d'un mot	VF0 à VF255
Octet (Byte)	0 à 255	1 octet	VB0 à VB255
Entier (Integer)	0 à 65535	2 octets non signés	VI0 à VI255
Entier long (Long)	+/- 2 147 483 647	4 octets signés	VL0 à VL255
Réel (Real)	+/- 2 147 483 647	4 octets signés	VR0 à VR255

Il est possible d'utiliser des variables indexées pour se déplacer dans un tableau.

VL22 = VL0[7] 'est équivalent VL22 = VL7

VL23 = VL2[9] 'est équivalent VL23 = VL11

VB3 = 9

VL24 = VL5[VB3] 'est équivalent VL24 = VL14

Les variables du type réel sont des entiers longs signés que l'on divise par un coefficient du type 1, 0.1, 0.01 … (type réel à virgule fixe)

Pour changer ce coefficient, aller dans **Motion Control -> Configuration -> Unités -> Précision** (on peut également le modifier en passant par **Option -> Langage DPL -> Compilateur**)

6-2-2- Conversions de type de données

Pour convertir un type de données en un autre, il suffit de faire une affectation :

- **Type flag :**

VB1 = VF0

VI1 = VF0

VL1 = VF0

VR1 = VF0

- **Type octet**

VF2 = VB0 ‘ VF2 est égale au 1er bit de poids faible de VB0

VI2 = VB0

VL2 = VB0

VR2 = VB0

- **Type entier**

VF3 = VI0 ‘ VF3 est égale au 1er bit de poids faible de VI0

VB3 = VI0 ‘ VB3 est égale aux 8 premiers bits de poids faible de VI0

VL3 = VI0

VR3 = VI0

- **Type entier long**

VF4 = VL0 ‘ VF4 est égale au 1er bit de poids faible de VL0

VB4 = VL0 ‘ VB4 est égale aux 8 premiers bits de poids faible de VI0

VI4 = VL0 ‘ VI4 est égale aux 16 premiers bits de poids faible de VL0

VR4 = VL0

- **Type réel**

VF5 = VR0 ‘ VF5 est égale au 1er bit de poids faible de la partie entière de VR0

VB5 = VR0 ‘ VB5 est égale aux 8 premiers bits de poids faible de la partie entière de VR0

VI5 = VR0 ‘ VI5 est égale aux 16 premiers bits de poids faible de la partie entière de VR0

VL5 = VR0 ‘ VL5 est égale à la partie entière de VR0

6-2-3- Notation numériques

Les valeurs numériques peuvent être exprimées en décimal, en hexadécimal, en binaire.

Exemple :

VB0=254	‘ notation décimale
VB1=0FEh	‘ notation hexadécimale
VB2=11111110b	‘ notation binaire

6-3- Les Tâches

6-3-1- Principes du multitâches

Le moniteur temps réel multitâches gère jusqu’à 4 tâches en parallèle :

Le multitâche bascule de la tâche courante vers la tâche suivante si :

↳ Le temps passé dans la tâche dépasse le *temps de vieillissement*. Ce temps est paramétrable à partir du menu Options / Langage DPL / Compilateur. Il est nécessaire de recompiler les tâches après une modification.

↳ rencontre d’une instruction bloquante :

- ⇨ Wait, Delay
- ⇨ Mova, Movr, Stop, Home

↳ rencontre de l’instruction NEXTTASK

En règle générale, une tâche courte permettra de traiter des événements plus rapides qu’une tâche longue.

6-3-2- Gestion des tâches

Chaque tâche possède un mode de démarrage qui a été paramétré lors de sa création :

↳ Démarrage automatique : à chaque démarrage du variateur, la tâche est lancée automatiquement.

↳ Démarrage manuel : la tâche n’est pas lancée automatiquement.

Un projet doit au moins contenir une tâche avec démarrage automatique. Il est conseillé d’avoir une seule tâche dans laquelle on écrit toute la partie initialisation de l’application et ensuite on lance les autres tâches.

On dispose de 5 instructions pour gérer les tâches :

↳ Run : lancement d’une tâche qui est à l’arrêt.

↳ Suspend : suspension (pause) d’une tâche en cours d’exécution.

- ✎ Continue : reprise de l'exécution d'une tâche suspendue là où elle c'était arrêtée.
- ✎ Halt : arrêt d'une tâche en cours d'exécution.
- ✎ Status : indique l'état de la tâche.

Exemple :

Tâche 1	Tâche 2
Prog	Prog
.....	.....
Run 2	If VR1 = 0 Halt 2
Wait Status(2)=0	.....
....	End Prog
End Prog	

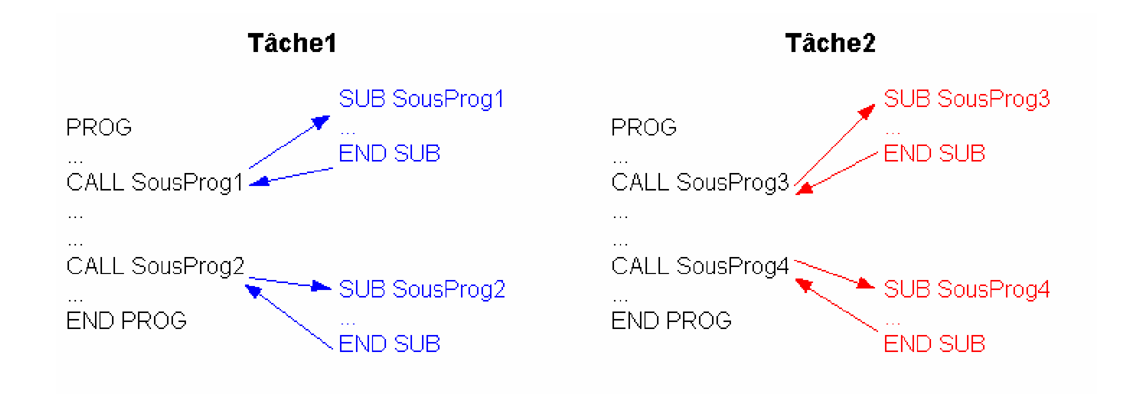
Attention : L'arrêt ou la suspension de la tâche n'affecte pas les mouvements lancés par celle-ci

Exemple :

Tâche 1	Tâche 2
Prog	Prog
.....	.....
If VF=0 Goto CYCLE_PROD	Mova(1000)
Halt 2	Out(6)=1
Stop	Mova(2000)
CYCLE_PROD	.....
....	End Prog
End Prog	

6-3-3- Structure d'une tâche basic

Chaque tâche est constituée d'un programme principal défini par les mots clé PROG et END PROG et par des sous programmes sous forme de structure SUB .. END SUB. Par exemple :



- **Programme principal**

Le programme principal d'une tâche peut appeler tous les sous programmes de la tâche mais ne peut pas appeler les sous programmes d'une autre tâche. Une tâche correspond à un fichier. Dans l'exemple précédent, la tâche1 peut appeler les sous-programmes SousProg1 et SousProg2 mais ne peut pas appeler les sous- programmes SousProg3 et SousProg4. Un sous programme d'une tâche peut également appeler un autre sous-programme de la même tâche.

Une seule structure PROG ... END PROG doit être utilisée par tâche. Elle peut apparaître à n'importe quel endroit.

Pendant l'exécution de la tâche, la rencontre du mot clé END PROG provoque un branchement de celle-ci en PROG.

- **Sous-programmes**

Un sous-programme doit être déclaré par une procédure SUB...END SUB. Il peut être placé avant ou après le programme principal.

Pour appeler un sous-programme, vous devez utiliser la fonction CALL. Le sous-programme appelé doit être dans la même tâche.

Après l'appel du sous-programme, son exécution et son retour, la tâche continue automatiquement à l'instruction qui suit l'appel du sous-programme. Le système sort d'un sous programme lorsqu'il rencontre l'instruction END SUB ou EXIT SUB. Par exemple :

```
SUB Calcul

VR2=0

IF VR1=0 EXIT SUB          ' Si VR1 est égal à zéro la division est impossible

VR2=VR10/VR1    ' Division

END SUB
```

Un sous-programme peut être appelé partout dans le programme mais ne peut s'appeler lui-même. Si des données sont utilisées dans le programme et dans des sous programmes, il est recommandé d'utiliser des variables bien spécifiques. En fait, toutes les variables

peuvent être modifiées par un sous-programme, vous pouvez donc utiliser ces variables spécifiques dans chaque sous-programme en les affectant simplement avant l'appel. Par exemple :

```
...
VR100=VR1
VR101=VR18
CALL Divise
IF VR102>10 Goto ...
...

SUB Divise
VR102=0
IF VR100=0 EXIT SUB
VR102=VR100/VR101
END SUB
```

- **Branchement à une étiquette**

  L'instruction GOTO sert à effectuer un saut à une adresse représentée par une étiquette. Une étiquette est composée d'un nom terminé par ":". Si l'instruction GOTO se trouve à l'intérieur d'une structure de sous-programme SUB...END SUB, l'étiquette doit se trouver dans cette même structure.

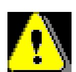
  Un branchement avec l'instruction GOTO peut être effectué indifféremment vers l'avant ou l'arrière du programme. Par exemple:

```
GOTO Label1
...
Label1:
...
```

- **Opérateurs**

  Les expressions sont composées d'opérateurs et d'opérandes. En Basic presque tous les opérateurs sont binaires, c'est à dire qu'ils utilisent deux opérandes. Les opérateurs n'utilisant qu'un opérande sont qualifiés d'unaires. Les opérateurs binaires utilisent les formes algébriques communes, par exemple A + B. Les opérateurs unaires s'écrivent toujours avant leurs opérandes, par exemple : `NOT A`. Dans des expressions complexes les règles de priorité suivantes enlèvent toute ambiguïté sur l'ordre des opérateurs.

Opérateur	Priorité	Type
NOT	Première (Haute)	Unaire
*, /, DIV, MOD, ,AND , <<, >>	Seconde	Multiplication
+, -, OR, XOR	Troisième	Addition
=, <>, <, >, <=, >=	Quatrième (Basse)	Comparaison

 **Dans une ligne programme, un seul opérateur pourra être traité.**

- **Opérateurs arithmétiques**

L'opérateur 'NOT' est un opérateur unaire. Les opérateurs + et - sont employés comme des opérateurs unaires ou des opérateurs binaires. Les autres sont uniquement binaires.

Un opérateur unaire ne possède qu'un paramètre.

Par exemple : NOT <Expression>

Un opérateur binaire demande deux paramètres.

Par exemple : <Expression1> * <Expression2>

- **Opérateurs binaires :**

Opérateur	Opération	Type de l'opérande	Type
+	Addition	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Soustraction	Octet, Entier, Entier long ou réel	Type de l'opérande
*	Multiplication	Octet, Entier, Entier long ou réel	Type de l'opérande
/	Division	Octet, Entier, Entier long ou réel	Réel
DIV	Division d'entiers	Octet, Entier, Entier long	Type de l'opérande
MOD	Modulo	Octet, Entier, Entier long	Type de l'opérande

- **Opérateurs unaires :**

Opérateur	Opération	Type de l'opérande	Type
+	Même signe	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Inversion de signe	Octet, Entier, Entier long ou réel	Type de l'opérande

- **Opérateurs logiques :**

Opérateur	Opération	Type de l'opérande	Type
NOT	Négation binaire	Octet, Entier	Type de l'opérande
AND	ET logique	Octet, Entier	Type de l'opérande
OR	OU logique	Octet, Entier	Type de l'opérande
XOR	OU exclusif	Octet, Entier	Type de l'opérande
>>	Décalage à droite	Octet, Entier	Type de l'opérande
<<	Décalage à gauche	Octet, Entier	Type de l'opérande

- **Opérateurs sur bits :**

Opérateur	Opération	Type de l'opérande	Type
+	Même signe	Octet, Entier, Entier long ou réel	Type de l'opérande
-	Inversion de signe	Octet, Entier, Entier long ou réel	Type de l'opérande

- **Opérateurs de relation :**

Opérateur	Opération	Type de l'opérande	Type
=	Egal	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<>	Différent de	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<	Inférieur à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
>	Supérieur à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
<=	Inférieur ou égal à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit
>=	Supérieur ou égal à	Octet, Entier, Entier long, Réel ou chaîne de caractères	Bit

- **Tests**

Les instructions conditionnelles sont un moyen pratique d'exécuter ou non un groupe d'instructions selon qu'une condition est vraie ou fausse :

IF <Expression> GOTO <Etiquette>

…

Etiquette:

…

<Expression> doit être une valeur de type bit. Si <Expression> est vraie alors un saut à <Etiquette> est exécuté. Si <Expression> est fausse, le programme passe directement à la ligne suivante.

Exemple :

```
VEL%=100                              ' Vitesse rapide
STTA=2000                             ' Départ de l'axe à la position absolue 2000
MOVE_ON:
IF POS _S<1000 GOTO SUITE_VEL         'Si la position est supérieure ou égale à 1000 alors
    VEL%=50                                  ' la vitesse est diminuée de moitié.
SUITE_VEL:
    IF POS_S<1500 GOTO SUITE_OUT             'Si la position est supérieure ou égale à 1500 alors
    OUT(9)=1                          'la sortie 9 est activée.
    SUITE_OUT:
IF MOVE_S<>1 GOTO MOVE_ON             'Reboucle tant que le mouvement n'est pas fini.
…
```

7- Programmation du contrôle de mouvement

7-1- Introduction

Le variateur peut gérer un axe servo et un codeur maître.

Le logiciel DPL contient de nombreuses instructions évoluées pour le contrôle de mouvement :

Positionnement, arbre électrique, superposition de mouvement...

Les limites du compteur de position sont de ± 2048 tours motrices

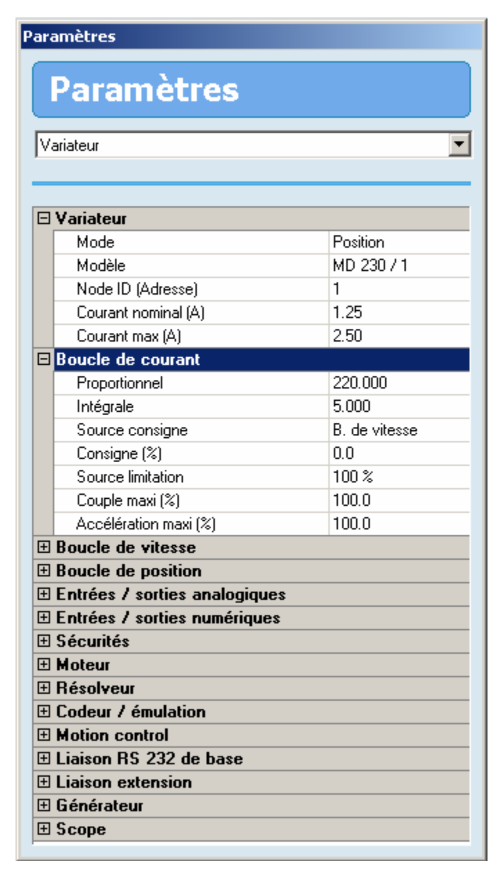
Il est possible d'inverser le sens moteur en boucle de position à partir de la liste de paramètres : Motion control / Inversion sens moteur (Attention, cela n'inverse pas pour autant la position du rotor moteur lue notamment au tableau de bord).

7-2- Paramétrage d'un axe

7-2-1- Réglage d'un axe

Un axe doit être paramétré avant de pouvoir l'utiliser.

L'accès aux paramètres se fait à partir du menu **Paramètre** ou par accès direct grâce à la fenêtre « paramètres ».



A) Régulation

Il est conseillé d'utiliser la bibliothèque de paramètre moteur afin de calibrer les boucles de régulation nécessaire au bon fonctionnement du moteur, pour plus d'information voir le chapitre 4.

B) Erreur de poursuite maxi

Dès qu'un axe passe en mode asservi, il est contrôlé à tout moment : à l'arrêt, en mouvement.

Si la différence entre sa position théorique calculée et sa position réelle donnée par le retour résolveur est supérieure à l'erreur de poursuite maxi, le système passe l'axe servo en mode non asservi et ouvre le contact de le sortie « variateur prêt » (sauf si utilisation de l'instruction SECURITY).

Le réglage de cette valeur est très important : une valeur trop petite entraîne des arrêts intempestifs sur l'axe, une valeur trop grande influe sur la sécurité des organes électriques et mécaniques.

Rentrer dans le champ «Erreur de poursuite maxi» de la fenêtre **Paramètre \ sécurités \ position**, la valeur adéquate (cette valeur est dans l'unité sélectionnée).

C) Fenêtre de position

Lorsque l'on envoie un axe à une position, la variateur considère que le mouvement est terminé quand le profil théorique de la trajectoire est exécuté et que la position réelle est comprise entre +/- la fenêtre de position. Par exemple, sur une machine de perçage où l'on recherche une précision de +/- 0.1mm, on réglera la fenêtre à cette valeur.

Rentrez dans le champ «Fenêtre de position » de la fenêtre **Paramètre \ sécurités \ position**, la valeur de précision recherchée (cette valeur est dans l'unité sélectionnée).

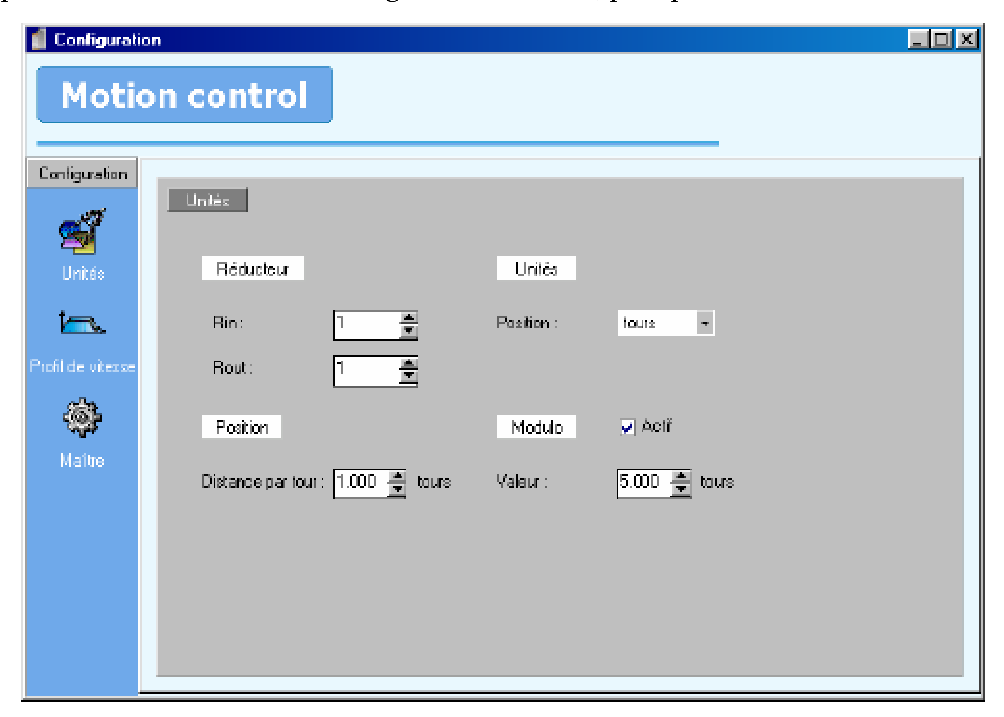
7-2-2- Unité utilisateur

En fonction de l'application, de la mécanique (axe linéaire, rotatif), il est d'intéressant de pouvoir affecter à chaque axe une unité utilisateur représentative : mm, point (point codeur * 4), degrés, radian, pouce, tour, unité quelconque...

En fait, l'unité est utilisée uniquement sur les écrans du DPL afin d'y apporter un confort d'utilisation et de compréhension.

Par exemple, si le choix de l'unité est « mm », dans l'écran de Configuration « Unités » du DPL, la vitesse sera exprimée en mm/s, les accélérations et décélérations en mm/s²...

Cliquer sur **Motion Control \ Configuration \ Unités**, pour paramétrer l'unité de votre axe :



Exemple 1 : Axe infini

Moteur en bout de vis à bille au pas de 5mm. Unités = mm, Rin = 1, Rout = 1, Distance par tour = 5.000, Modulo non activé

Exemple 2 : Axe infini

Moteur avec réducteur de 10. En sortie de réducteur, tourelle 360°, Unités = degrés, Rin = 10, Rout = 1, Distance par tour = 360.000, modulo activé avec une valeur de 360.000

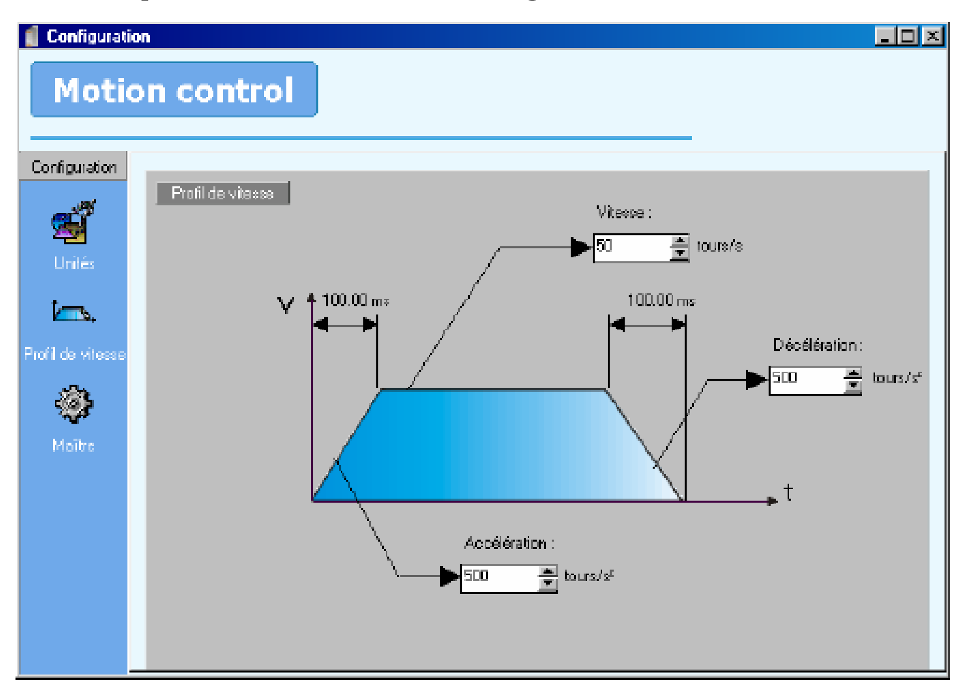
Nota : le nombre de chiffres après la virgule est paramétrable dans le menu *Options / Langage DPL*

7-2-3- Profil de vitesse

Une trajectoire en positionnement intègre les phases d'accélération, de vitesse plateau, de décélération.

Les champs contenus dans la configuration du variateur permettent de donner des valeurs par défaut à ces différentes phases. Les valeurs sont prises en compte à chaque démarrage du variateur, elles sont également utilisées par en mode trajectoire, par les outils de réglage : Motion et Générateur ainsi que par les instructions ACC%, DEC%, VEL%.

Cliquer sur **Motion Control \ Configuration \ Profil de vitesse**,:



La décélération urgente est utilisée pour arrêter le mouvement lorsqu'on utilise les entrées de Fin de course.

7-3- Mode asservi / non asservi

7-3-1- Passage en mode non asservi

L'axe passe en mode non asservi (boucle ouverte) :

- A chaque redémarrage du variateur.
- A chaque exécution de l'instruction AXIS OFF à partir d'une tâche.
- Sur erreur de poursuite de l'axe (sauf si l'instruction SECURITY a été affectée).
- Sur un forçage à partir des menus de debug (bouton *enable* en position OFF), du menu communication (arrêt des tâches, redémarrage des tâches, Envoyer les tâches).

L'instruction AXIS_S permet de lire l'état dans lequel se trouve l'axe.

Si une instruction de mouvement est envoyée alors que l'on est en boucle ouverte, elle sera consommée mais le mouvement ne sera pas effectué.

Par exemple :

Tâche Process

```
PROG
...
...            ' Le variateur a détecté une erreur de poursuite
...            ' => L'axe passe en mode non asservi
MOVA=1000      ' le mouvement est consommé mais non effectué
OUT(3)=1       ' Activation de la sortie n°3
MOVA=2000      ' le mouvement est consommé mais non effectué
OUT(3)=0       ' Désactivation de la sortie n°3
...            ' La sortie S1 est passée fugitivement à 1 car
...            ' L'instruction Mova=2000a pris peu de temps au système
END PROG
```

7-3-2- Passage en mode asservi

Pour que l'axe servo puisse piloter et contrôler les mouvements, il est nécessaire de le passer en mode asservi.

L'axe passe en mode asservi (boucle fermée) :

- A chaque exécution de l'instruction AXIS ON à partir d'une tâche.
- Sur un forçage à partir des menus de debug (bouton *enable* en position ON).

L'instruction AXIS_S permet de lire l'état dans lequel se trouve l'axe.

La prise en compte de l'instruction Axis est effectuée au bout d'environ 3ms. Pour s'assurer que l'asservissement est effectif, écrire :

```
Axis On
Wait AXIS_S=On
```

7-4- Prise d'origine

7-4-1- Définition :

La Prise d'origine permet au système de déterminer l'origine mesure de l'axe, celle-ci étant perdue à chaque coupure d'alimentation.

La prise d'origine machine (P.O.M) permet de référencer la position moteur par rapport à une position de la mécanique.

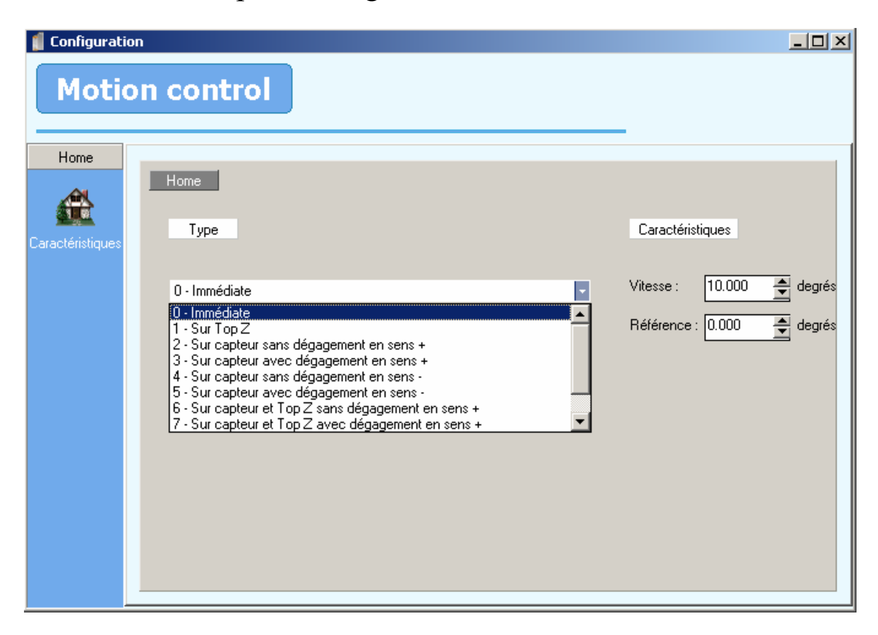
Différents types de POM sont disponibles : immédiat, sur capteur, avec dégagement.

Un cycle de POM force le compteur de position moteur à une valeur de référence.



7-4-2- Configuration de la POM sous DPL :

Pour accéder au paramétrage de la POM, aller dans **Motion control \ Home**



A partir de cet écran, on configure le type de POM, la vitesse et position de référence à charger dans le compteur de position.

Informations :

- Le type choisi dans cet écran est utilisé uniquement sur un mouvement HOME déclaré à partir du tableau Trajectoires lorsque le variateur travaille en mode « trajectoires préenregistrées »
- Si on utilise l'instruction HOME dans une tâche basic, le type doit être indiqué dans l'instruction.

  Exemple : de POM sur top Z -> HOME (1)

- La vitesse de l'axe pendant la POM correspond à la vitesse saisie dans cet écran. Si pendant la POM, l'instruction VEL ou VEL% est exécutée, la vitesse de l'axe est alors modifiée.
- L'instruction Home est bloquante pour la tâche DPL. Si l'on souhaite arrêter une POM en cours d'exécution, il faut à partir d'une autre tâche : faire un HALT de la tâche contenant l'instruction HOME, puis un STOP de l'axe.

7-4-3- Les types de POM :

A) Type 0 : immédiate :

Le compteur de position est forcé à la valeur de référence de façon immédiate.

Exemple : Référence = 100 dans la fenêtre de saisie

HOME (0) ‘ position moteur = 100

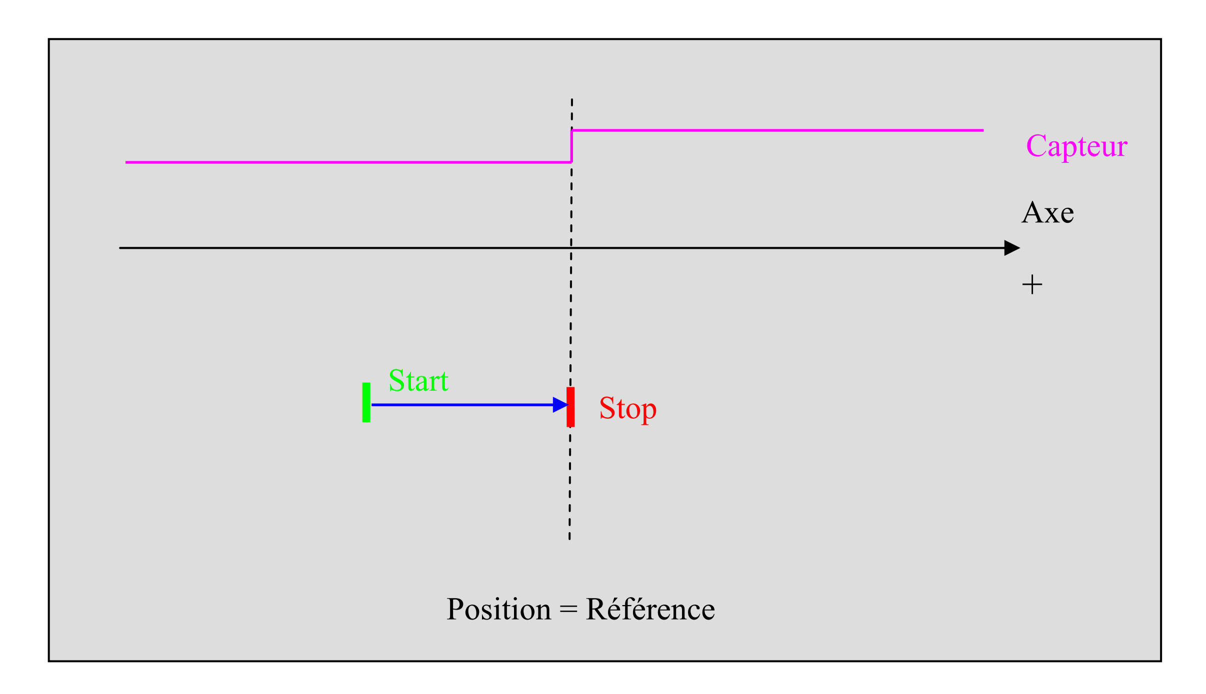
B) Type 1 : sur TOP Z :

Le moteur n'effectue aucun déplacement mais sa position est recalculée par rapport au Top Z moteur et à la valeur de référence. On obtient une position se situant entre +/- ½ tour ou référence +/- ½ tour moteur.

C) Type 2 : Sur capteur, en sens +, sans dégagement

Le variateur lance un mouvement infini en sens + et attend un front montant sur l'entrée HOME.

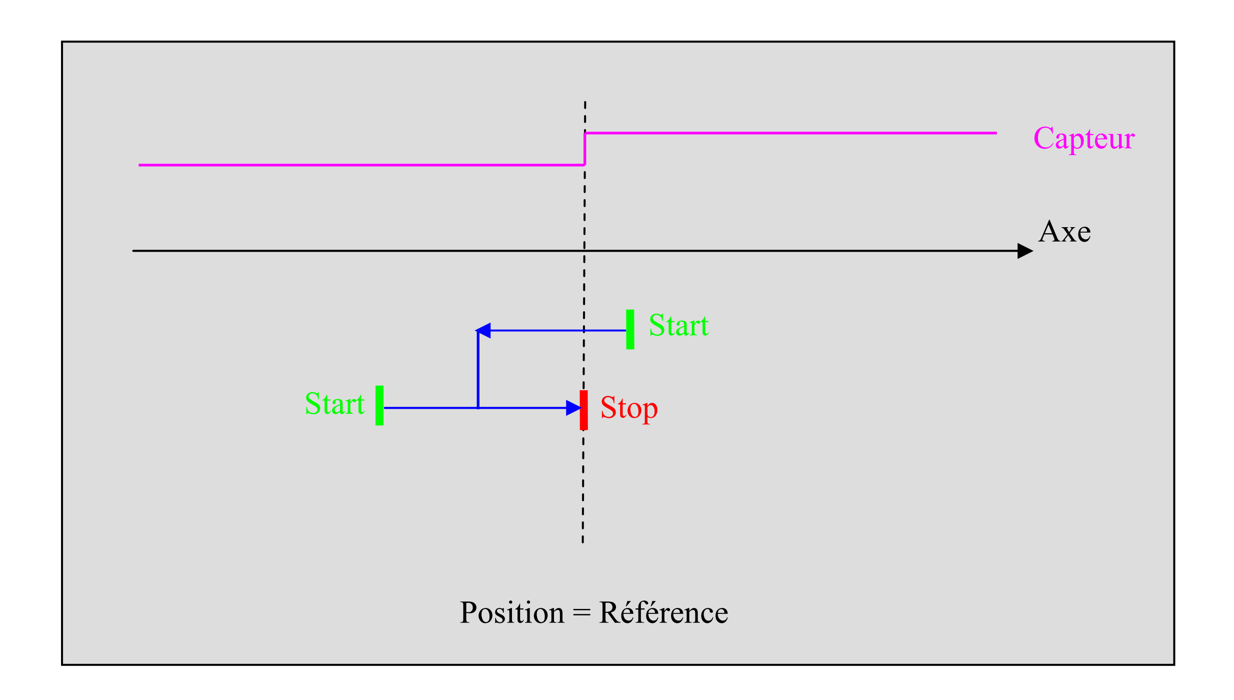
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



D) Type 3 : Sur capteur, en sens +, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens – pour se dégager du capteur HOME.

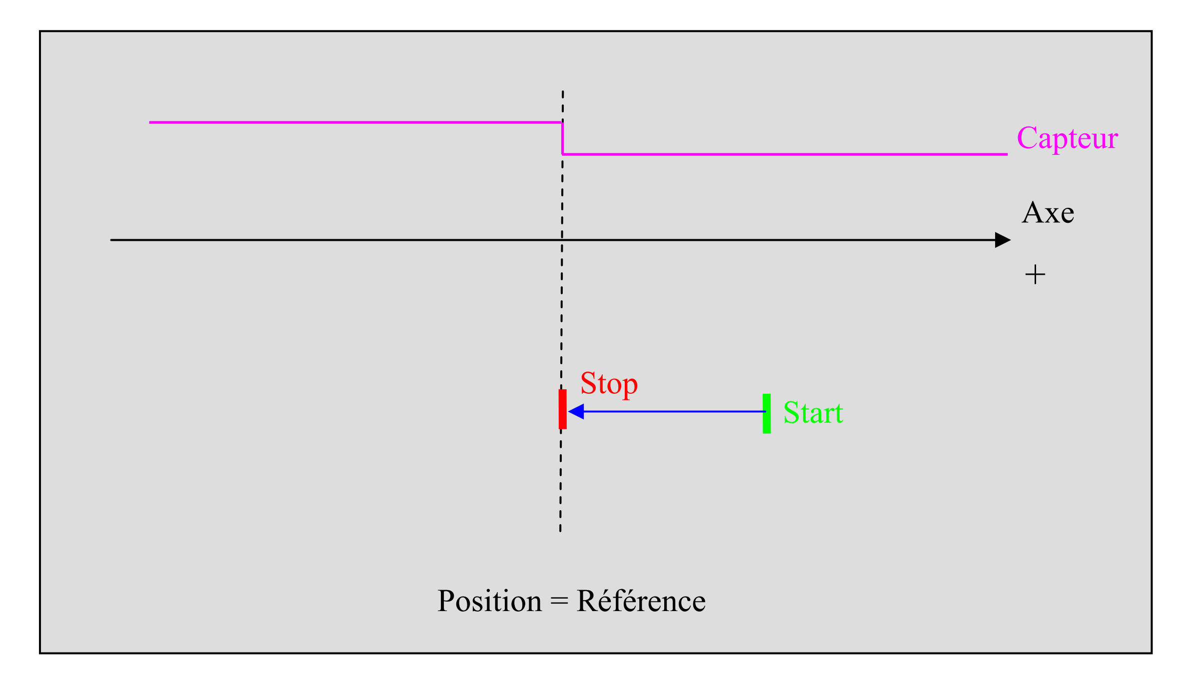
Le variateur lance ensuite un mouvement infini en sens + et attend un front montant sur l'entrée HOME pour forcer la position à la valeur de référence et le moteur s'arrête à cette position.



E) Type 4 : Sur capteur, en sens -, sans dégagement

Le variateur lance un mouvement infini en sens - et attend un front montant sur l'entrée HOME.

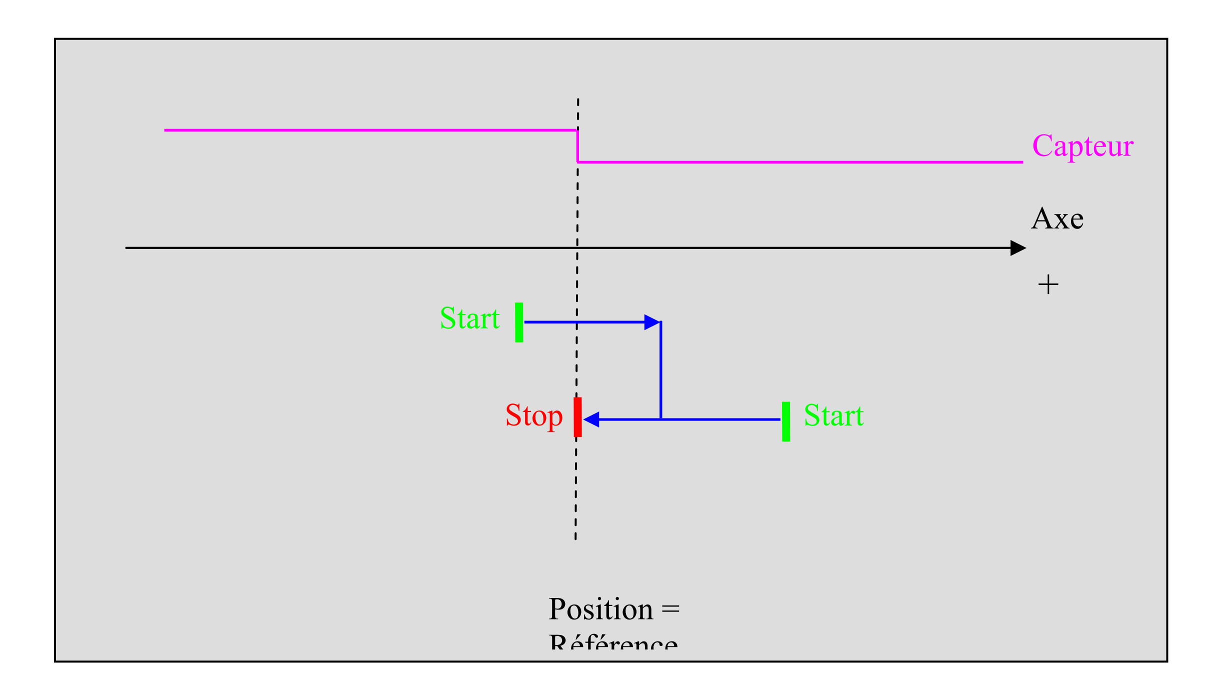
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



F) Type 5 : Sur capteur, en sens -, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens + pour se dégager du capteur HOME.

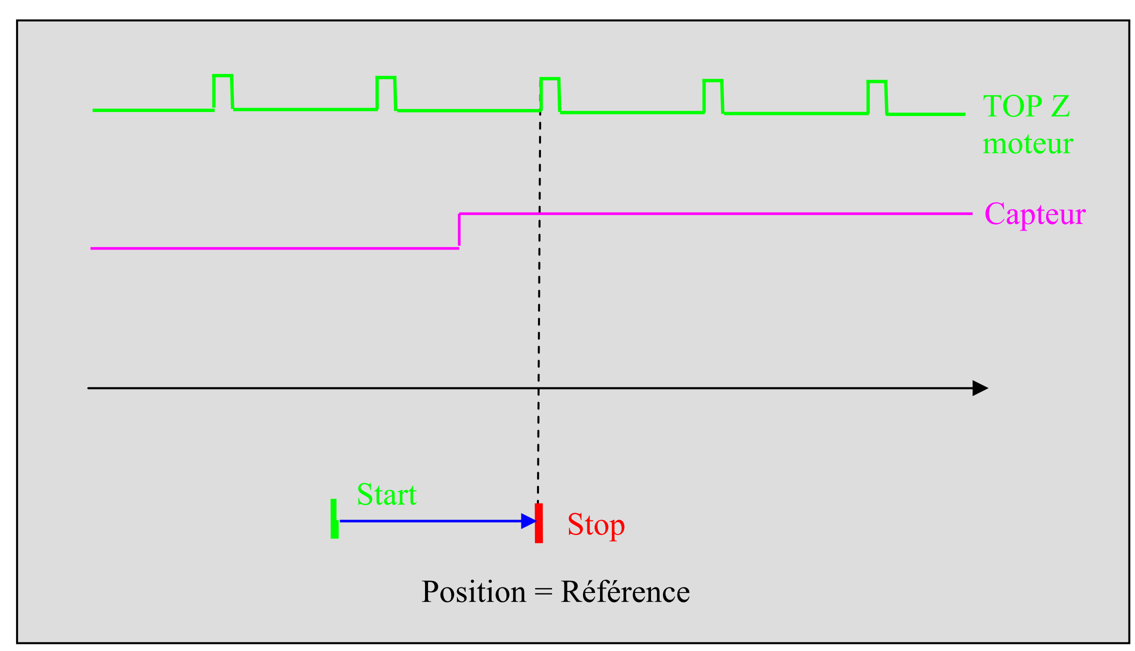
Le variateur lance ensuite un mouvement infini en sens - et attend un front montant sur l'entrée HOME pour forcer la position à la valeur de référence et le moteur s'arrête à cette position.



G) Type 6 : Sur capteur et TOP Z, en sens +, sans dégagement

Le variateur lance un mouvement infini en sens + et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.

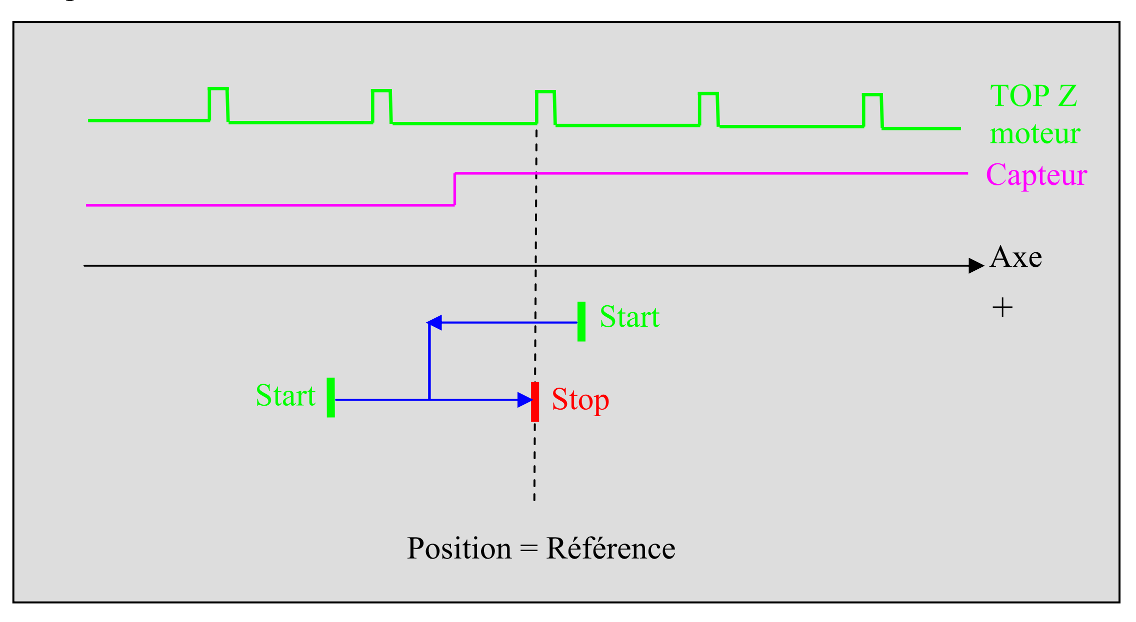


H) Type 7 : Sur capteur et TOP Z, en sens +, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens - pour se dégager du capteur HOME.

Le variateur lance ensuite un mouvement infini en sens + et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

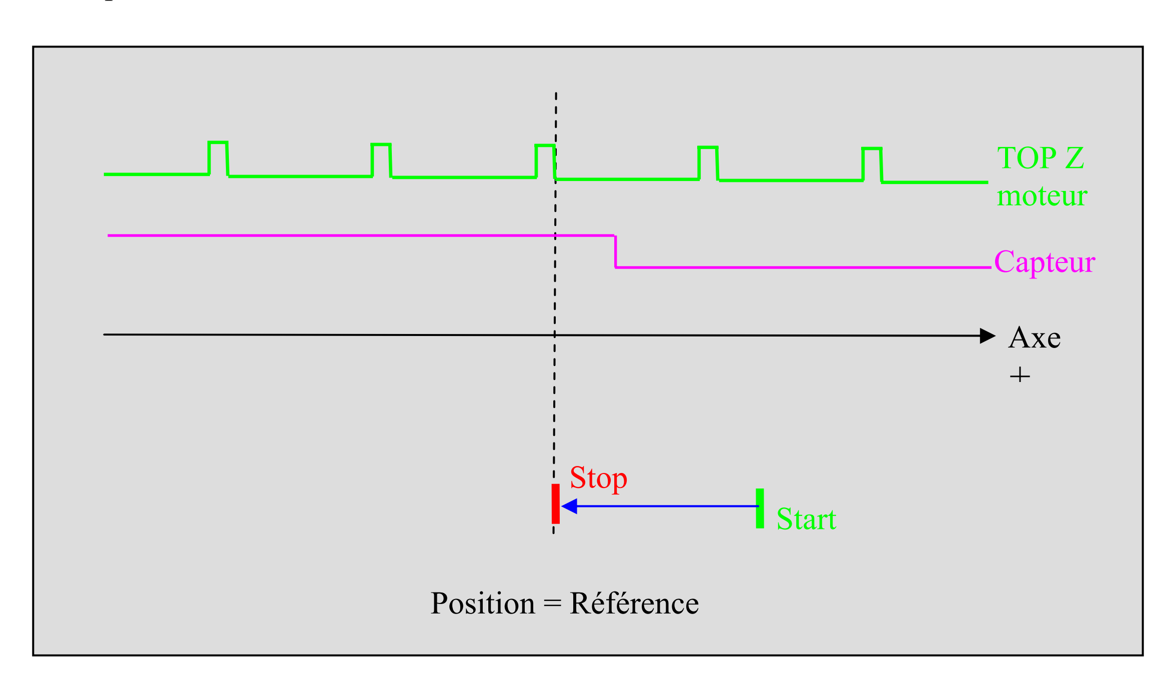
La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



I) Type 8 : Sur capteur et TOP Z, en sens –, sans dégagement

Le variateur lance un mouvement infini en sens - et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.

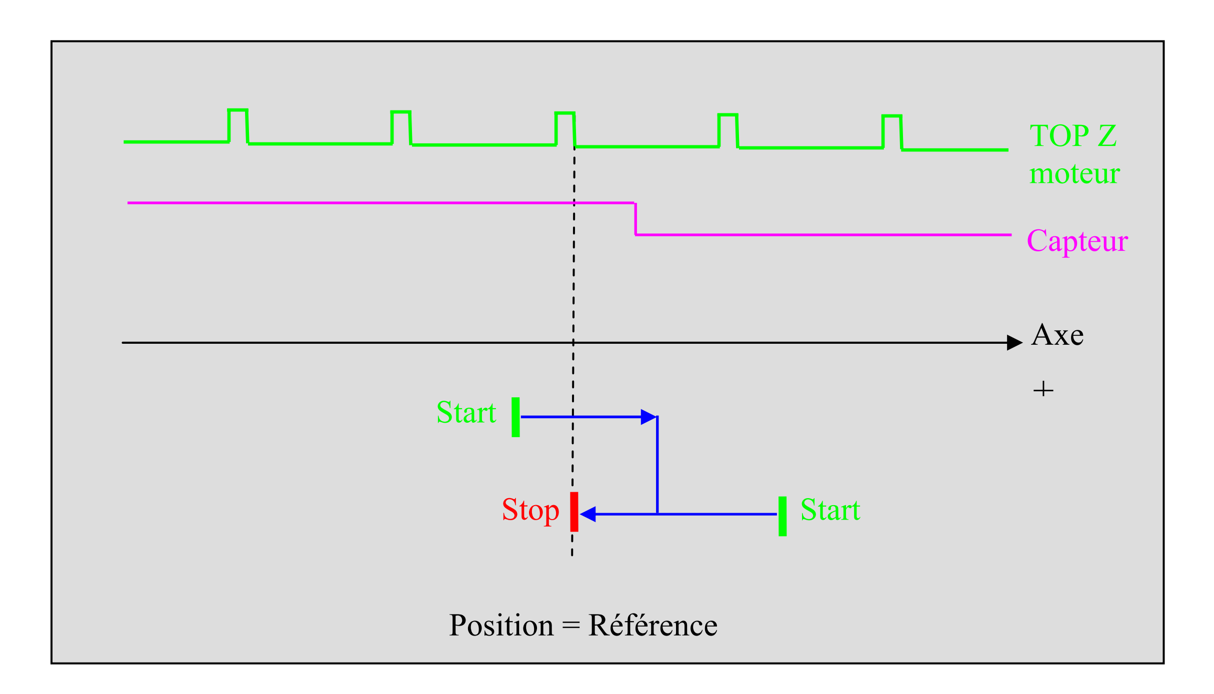


J) Type 9 : Sur capteur et TOP Z, en sens -, avec dégagement

Si l'entrée HOME est déjà à 1 alors le variateur lance en premier un mouvement infini en sens + pour se dégager du capteur HOME.

Le variateur lance ensuite un mouvement infini en sens - et attend un front montant sur l'entrée HOME puis le passage par le TOP Z moteur.

La position est alors forcée à la valeur de référence et le moteur s'arrête sur cette position.



7-5- Déclaration d'un axe en mode virtuel

A partir d'une tâche DPL, il est possible de faire passer un axe en mode virtuel grâce à l'instruction LOOP On. Dans ce mode, le variateur simulera le retour résolveur de façon interne et ainsi tous les mouvements seront exécutés virtuellement.

Ce mode est intéressant lors de la phase développement du programme : on peut tester la globalité de l'application sans avoir les variateurs et moteurs connectés.

Dans ce mode, ne pas brancher la puissance sur le connecteur X10

L'instruction LOOP OFF permet de désactiver le mode virtuel.

7-6- Positionnement

7-6-1- Mouvements absolus

- **Départ de mouvement : STTA**

Pour lancer un mouvement vers une position absolue et ne pas attendre sa fin pour poursuivre l'exécution de la tâche, on doit utiliser STTA. Cette instruction est très utile si la vitesse ou la position à atteindre doit changer en cours de mouvement. Avec cette fonction, l'erreur absolue est minimale.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

STTA=Position

Par exemple :

```
VEL%=100
STTA=2000                  ' Départ de l'axe à la position absolue 2000
WAIT POS_S >200                  ' Attente position 200
OUT (6)=1                  ' Activation d'une sortie
WAIT POS_S >700                  ' Attente position 700
OUT (6)=0                  ' Désactivation de la sortie
WAIT MOVE_S=0        ' Attente fin de mouvement
```

Dans cet exemple, pendant le mouvement, on peut changer des sorties car la tâche n'est pas bloquée.

Si l'instruction MERGE est activée et que l'on charge plusieurs STTA, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle.

Si l'axe est modulo, un lancement à une position sera effectué dans le sens positif si la valeur demandée est positive, sens négatif dans le sens contraire. Par exemple :

Axe modulo 360°

Axe en position initiale à 90°

```
STTA=-10 'déplacement dans le sens – d'une distance de 80°
WAIT MOVE_S=0
STTA=350 'déplacement dans le sens + d'une distance de 340°
WAIT MOVE_S=0
STTA=20  'déplacement dans le sens + d'une distance de 30°
WAIT MOVE_S=0
```

- **Mouvement : MOVA**

La fonction MOVA envoie l'axe à une position absolue. Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

MOVA=Position

Cette fonction envoie l'axe à la position absolue dont la valeur est <Position>. Le programme attend la fin du mouvement avant de continuer. L'erreur de positionnement absolue est minimale.

Par exemple :

```
MOVA=100
CALL Percage
MOVA=0
```

L'instruction MOVA est bloquante pour la tâche tant que le mouvement n'est pas terminé (condition MOVE_S=0).

MOVA=100 est équivalent à STTA=100

WAIT MOVE_S=0

- **Trajectoire : TRAJA**

La fonction Trajectoire est conçue pour simplifier la définition de mouvements complexes.

Elle permet de lancer un mouvement vers une position absolue, avec une vitesse spécifique.

Syntaxe de la fonction TRAJ :

TRAJA (<Positon>, <Vitesse>)

Par exemple :

TRAJA (500,2000)

Cet exemple est équivalent à :

VEL=2000

STTA = 500

Si l'instruction MERGE est activée et que l'on charge plusieurs TRAJA ou TRAJR, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle. Par exemple :

MERGE On

TRAJA(500,2000)

TRAJA(1000,50) ‘passage en petite vitesse à la position 500

7-6-2- Mouvements relatifs

- **Départ de mouvement : STTR**

Pour lancer un mouvement vers une position relative et ne pas attendre sa fin pour poursuivre l'exécution de la tâche, on doit utiliser STTR. Cette instruction est très utile si la vitesse ou la position à atteindre doit changer en cours de mouvement.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

STTR (Position)

Par exemple :

```
VEL%=100                    ' Vitesse rapide
VR1=POS_S
STTR=2000                   ' Départ de l'axe à la position relative 2000
BOUCLE:
VR2=POS_S
VR2=VR2-VR1
IF VR2 < 100 GOTO BOUCLE    ' Attente position +100
VEL%=10                     ' Vitesse lente
WAIT MOVE_S=0        ' Attente fin de mouvement
```

Dans cet exemple, pendant un mouvement, la vitesse peut être modifiée car l'exécution du programme n'est pas bloquée.

Si l'instruction MERGE est activée et que l'on charge plusieurs STTR dans le variateur, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle.

- **Mouvement : MOVR**

La fonction MOVR envoie l’axe à une position relative. Elle utilise les valeurs courantes d'accélération, de décélération et de vitesse. La syntaxe est :

MOVR=Distance

Cette fonction envoie l’axe à une valeur relative <Distance>. Le programme attend la fin du mouvement avant de continuer.

Par exemple :

```
VB1=0
BOUClE:
  MOVR=100
  CALL PERCAGE
  VB1=VB1+1
  IF VB1<10 Goto BOUCLE
```

L’instruction MOVR est bloquante pour la tâche tant que le mouvement n’est pas terminé (condition MOVE_S=0).

MOVR=100 est équivalent à STTR=100
WAIT MOVE_S=0

- **Trajectoire : TRAJR**

La fonction Trajectoire est conçue pour simplifier la définition de mouvements complexes.

Elle permet de lancer un mouvement vers une position relative, avec une vitesse spécifique.

Syntaxe de la fonction TRAJ :

TRAJR (<Positon>, <Vitesse>)

Par exemple :

TRAJR (500,2000)

Cet exemple est équivalent à :

VEL=2000

STTR = 500

Si l'instruction MERGE est activée et que l’on charge plusieurs TRAJA ou TRAJR, les mouvements seront exécutés les uns après les autres sans passer par une vitesse nulle. Par exemple :

MERGE On

TRAJR(500,2000)

TRAJR(1000,50) ‘passage en petite vitesse à la position 500 et arrêt du mouvement à la position 1500.

7-6-3- Mouvements infinis

Pour lancer un mouvement continu, il faut utiliser l'instruction STTI. L'axe se déplace alors à sa vitesse courante.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).

L'instruction STOP ou SSTOP est nécessaire pour arrêter un mouvement continu. Le sens de déplacement est défini par le caractère "+" ou "-".

Syntaxe :

STTI (Signe)

Exemple :

```
WAIT  INP(4)=On
STTI(+)
WAIT  INP(4)=Off
STOP
```

7-6-4- Arrêt d'un mouvement

Pour arrêter un mouvement, il faut utiliser les instructions STOP ou SSTOP. Elles arrêtent l'axe via leur décélération programmée et elles vident le buffer de mouvement.

L'instruction STOP est bloquante pour la tâche tant que le mouvement n'est pas terminé (condition MOVE_S=0) alors que SSTOP n'est pas bloquante.

Syntaxe : STOP

Exemple : déplacement jusqu'à un capteur.

```
STTI(+)
WAIT  INP(4)=On
STOP
```

L'instruction AXIS OFF arrête aussi le mouvement mais sans aucun contrôle car l'asservissement est inhibé.

7-7- Synchronisation

7-7-1- Arbre électrique :

- **GEARBOX :**

Cette instruction permet de réaliser un arbre électrique entre un codeur et le moteur (axe esclave).

Syntaxe :

GEARBOX (<Numérateur>, <Dénominateur>)

<Numérateur> / <Dénominateur> définit le rapport entre un tour du moteur esclave et un tour du codeur. En fait, pour [Résolution codeur maître*<Dénominateur>] incréments, le moteur va bouger de [4096*<Numérateur>] incréments, sachant que l'on à 4096 incréments par tour moteur.

Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).Tant que la liaison entre le maître et l'esclave ne sera pas coupée, l'instruction MOVE_S sera égale à 1 (même si l'esclave est arrêté).

L'instruction GEARBOX initialise de façon interne la valeur de GEARBOXRATIO à 4096.

Exemple : Si Numérateur = 1 et Dénominateur = 2, pour 1 tour codeur maître, le moteur esclave tourne de 0.5 tour.

Le Numérateur doit être inférieur ou égal à 8 et de type entier.

Le dénominateur doit être de type entier et doit respecter : (dénominateur*Résolution du codeur maître) <= 32768

Exemple : Codeur maître 4000 incréments -> Dénominateur doit être inférieur à 8.

Un gearbox avec des valeurs <Numérateur> ou <Dénominateur> différentes de 1, affecte la mise à l'échelle de la position du codeur maître (à prendre en compte si l'on traite cette position ou une boîte à contact sur l'axe maître).

- **STARTGEARBOX :**

Cette instruction permet de lancer un arbre électrique suivant une accélération et un rapport défini précédemment par GEARBOX. Le rapport entre le maître et l'esclave est : (ratio × <Numérateur>) / (<Dénominateur> × 4096), avec <Numérateur> et <Dénominateur> définis dans l'instruction GEARBOX.

Syntaxe : STARTGEARBOX (<Accélération>)

<Accélération> de 0 à 65535

La phase d'accélération sera est : (Ratio × 640µs) / Accélération

Avec Ratio correspondant à la valeur de GEARBOXRATIO.

- **GEARBOXRATIO :**

Cette instruction permet de modifier le rapport de réduction d'une liaison arbre électrique.

L'instruction STARTGEARBOX ayant déjà été exécutée.

Syntaxe : GEARBOXRATIO (<Ratio>)

<Ratio> de 0 à 32767: le rapport de l'arbre est défini par (<Ratio> × <Numérateur>) / (<Dénominateur> × 4096). <Numérateur> et <Dénominateur> sont les paramètres de l'instruction GEARBOX. En fait, pour [Résolution codeur maître*<Dénominateur>] incréments, le moteur va bouger de [Ratio*<Numérateur>] incréments, sachant que l'on à 4096 incréments par tour moteur.

L'instruction est non bloquante et permet de changer de ratio sans arrêter l'arbre électrique.

Un GEARBOXRATION n'affecte pas la mise à l'échelle de la position du codeur maître.

L'exécution de l'instruction GEARBOX initialise de façon interne le <Ratio> de GEARBOXRATIO à 4096.

- **STOPGEARBOX :**

Cette instruction permet d'arrêter un arbre électrique suivant une décélération.

Syntaxe :

STOPGEARBOX (<Décélération>)

<Décélération> de 0 à 65535

La phase de décélération sera de : (Ratio × 640µs) / Décélération

Avec Ratio correspondant à la valeur de GEARBOXRATIO

Après l'instruction STOPGEARBOX, il faut recopier la position réelle dans la position théorique car cette dernière n'as plus évolué à cause du l'arbre électrique.

- **Exemple :**

GEARBOX (1, 2)	'Le moteur tourne 2 fois moins vite que le codeur
GEARBOXRATION (4096)	
…	
STARTGEARBOX (4)	'Lance l'arbre électrique avec une phase d'accélération
…	'De (4096 × 640 / 4) = 655360µs soit 0,65s
GEARBOXRATIO (3087)	'Ratio : (3687 × 1) / (2 × 4096) = 0.45
…	
STOPGEARBOX (2)	'Arrêt de l'arbre électrique avec une phase
WAIT MOVE_S=0	'de décélération de (3687 × 640 / 2)=1,18s
VR0=POS_S	'Stocke la position réelle de l'axe dans VRO
HOME (0, VR0)	'Copie la position réelle de l'axe dans la position théorique

7-8- Capture

7-8-1- Capture :

La capture permet d'enregistrer la position courante de l'axe sur un front d'une entrée variateur, la capture se faisant en moins de 640 µs.

- **CAPTURE1 ou CAPTURE2 :**

Les instructions CAPTURE1 et CAPTURE 2 sont utilisées pour enregistrer la position courante de l'axe.

Syntaxe : CAPTURE1 (<Source>, <N° de l'entrée>, < Fenêtre >, <Mini>, <Maxi>, <Intérieur>)

Avec cette instruction, le variateur attend un front sur l'entrée capture. Quand le front est détecté, la position est stockée dans la variable REGPOS1_S. Le flag REG1_S est alors positionné à vrai.

<Source> 0 pour codeur moteur, 1 pour entrée codeur maître.

<N° de l'entrée> numéro l'entrée sur laquelle on attend le front montant (de 1 à 16).

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l'axe est entre les positions <Mini> et <Maxi>.

<Intérieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

<Mini> doit toujours être inférieur à <Maxi>.

- **REG1_S ou REG2_S :**

Syntaxe : <VFx>=REG1_S

Description : Cette fonction indique si une capture de position a été effectuée.

Remarques : La valeur retournée n'est vraie qu'une fois par capture. REG1_S est remis automatiquement à 0 sur une opération de lecture. Sur une relance d'une autre capture et si REG1_S vaut 1 alors REG1_S est mis à 0.

- **REGPOS1_S ou REGPOS2_S :**

Syntaxe : <Expression>=REGPOS1_S

Types acceptés : Expression : réel

Description : Cette fonction retourne la dernière position capturée sur l'axe par l'exécution de l'instruction CAPTURE1.

- **Exemple :**

```
CAPTURE1 (0, 4, On, 10, 20, On)  'Capture position sur front montant de l’entrée 4,
                                 ‘ Lorsque l’axe du moteur est entre 10 et 20.
WAIT REG1_S = ON                 ‘Attente d’une capture
VR1 = REGPOS1_S                  ‘VR1 = valeur de la position lors de la capture
```

8- Programmation de l'automate

8-1- Entrées/Sorties logiques

8-1-1- Lecture des entrées

La fonction INP est utilisée pour lire 1 bit, INPB un bloc de 8 bits et INPW un bloc de 16 bits.

Les syntaxes sont : INP (<NuméroEntrée>), INPB (<NuméroBloc>), INPW

<NuméroEntrée> doit représenter le numéro d'une entrée et <NuméroBloc> le numéro d'un bloc de 8 entrées. Ce numéro correspond au numéro de l'entrée dans le module de configuration. Le type de données retournées est :

- Bit pour une entrée
- Octet pour un bloc de 8 entrées
- Entier pour un bloc de 16 entrées

Par exemple :

VF1= INP (3) 'lecture d'une entrée n°3

VB2 = INPB (1) 'lecture du premier bloc de 8 entrées

VB4 = INPB (2) 'lecture du deuxième bloc de 8 entrées

VI3= INPW 'lecture des 16 entrées

8-1-2- Ecriture des sorties

La fonction OUT est utilisée pour écrire 1 bit , OUTB un bloc de 8 bits.

Les syntaxes sont : OUT(<NuméroSortie>), OUTB(<NuméroBloc>).

<NuméroSortie>doit représenter le numéro d'une sortie ou <NuméroBloc> le numéro d'un bloc de 8 sorties. Ce numéro correspond au numéro dans le module de configuration. Le type de données utilisé est :

- Bit pour une sortie
- Octet pour un bloc de 8 sorties

Par exemple :

OUT(5) = 1 'Mise à 1 de la sortie n°5

OUTB(1) = 48 'écriture d'un bloc de 8 sorties

8-1-3- Lecture des sorties

Toutes les sorties peuvent également être lues. La valeur lue est la dernière valeur écrite. Cette caractéristique est très utile quand plus d'un programme utilise le même bloc de sorties. Donc, il est possible d'écrire seulement les sorties désirées dans une opération sans changer les autres.

Par exemple :

Pour mettre à 1 le quatrième bit d'un bloc de 8 bits :

```
OUTB(2)= 16        'mise à 1 du quatrième bit du bloc n°2 de 8 bits
VB0 = OUTB(2)
```

8-1-4- Attente d'un état

Il est possible d'attendre un changement d'état sur une entrée grâce à l'instruction WAIT.

La syntaxe est : WAIT <Condition>

La fonction WAIT est utilisée pour attendre une condition de changement durant une exécution normale. L'exécution est stoppée aussi longtemps que la condition est fausse. Quand l'état devient vrai, l'exécution continue. Cette fonction est très utile pour attendre la fin des mouvements ou une butée logicielle...

Exemple :

```
WAIT INP(2) = ON        'Attente que l'entrée 2 soit à 1
STOP                    'Arrêt de l'axe
WAIT INP(5) = ON        'Attente que entrée n°5 soit à 1
```

8-1-5- Test d'un état

Il est possible de tester l'état d'une entrée grâce à l'instruction IF…

La syntaxe est : IF (<Condition>) GOTO <Etiquette>

La structure IF… est utilisée pour tester une condition à un instant donné. La validation de la <Condition> permet de réaliser un branchement à une étiquette.

Exemple :

```
IF INP(5) = ON GOTO Suite_OK        'Test de l'état de l'entrée n°5,
                                    'si entrée à 1 saut en Suite_OK
```

8-2- Entrées/Sorties analogiques

8-2-1- Lecture d'une entrée

Les fonctions ADC(1) et ADC(2) sont utilisées pour lire 2 entrées analogiques. Les données retournées par la fonction sont toujours de type réel et comprises entre -10 et +10.

Par exemple:

```
VR1 = ADC(1)              'Lecture de l'entrée analogique 1
VR5 = ADC(2)              'Lecture de l'entrée analogique 2
```

8-2-2- Ecriture d'une sortie

La fonction DAC est utilisée pour écrire sur la sortie analogique.

La syntaxe est : DAC=<Expression réelle>

Les données utilisées par l'instruction sont toujours de type réel et comprises entre -10 et +10.

Par exemple :

```
DAC=5.0                   'Ecriture d'une valeur de consigne de 5 V
```

8-3- Temporisations

8-3-1- Attente passive

La fonction DELAY est utilisée pour établir une attente passive. Sa syntaxe est :

DELAY <Durée>

<Durée> est un entier exprimé en milliseconde. Il est recommandé d'utiliser cette fonction pour une longue attente passive car le programme en attente ne prend pas de temps processeur.

Avec cette fonction, le programme attend la durée indiquée.

Par exemple:

```
Debut:
    WAIT INP(5) = 1
    ...
    DELAY 5000            ' Délai de 5 secondes
    ...
```

GOTO Debut

8-3-2- Attente active

- **TIME :**

La variable globale interne TIME peut être utilisée pour établir des attentes actives. TIME est un entier long qui représente le nombre de 0.64 millièmes de seconde écoulées depuis la dernière mise sous tension. Cette variable peut donc être utilisée comme base de temps. Elle convient en particulier aux machines qui sont sous-tension moins de 16 jours. En effet à la mise sous-tension, TIME est initialisé à 0. Au-delà de16 jours, la variable atteint sa valeur maximum 2^31 et passe ensuite à 2^-31. Cette transition appelée débordement peut provoquer dans certains cas des erreurs de temporisations, pour éviter ce problème il est préférable d'utiliser l'instruction LOADTIMER.

Par exemple :

```
VL2=TIME
VL2=VL2 + 7812
ATTENTE :
 VL3=TIME
  IF VL3<VL2 GOTO ATTENTE                    'Temporisation de 5s
```

Remarque : TIME est de type entier long

Attention : La fonction TIME ne fonctionne pas dans un test

- **LOADTIMER et TIMER :**

L'instruction LOADTIMER peut être utilisée pour établir des attentes actives. C'est un réel qui représente le nombre de 0.64 millièmes de seconde écoulées depuis la dernière mise sous tension. Cette variable peut donc être utilisée comme base de temps. Elle convient en particulier aux machines qui sont toujours sous-tension.

Elle permet également de charger dans un timer une valeur, qui se décrémentera automatiquement jusqu'à 0. Il est possible de savoir si le timer est écoulé en utilisant l'instruction TIMER(VLXX), avec XX compris entre 0 et 255.

Si TIMER(VLXX) = 1 la temporisation n'est pas écoulée.

Si TIMER(VLXX) = 0 la temporisation est écoulée.

Il est possible d'utiliser simultanément 256 timers.

Par exemple :

```
LOADTIMER(VL129)=4688              'Chargement d'une temporisation de 3s
```

```
BOUCLE :
    IF TIMER(VL129)<>0 GOTO BOUCLE          'Attente de la fin de la tempo
```

Remarque : Pendant l'exécution de ces lignes la variable VL129 de type entier long est utilisée par le système

8-4- Compteurs

Attention :

- Une même entrée ne peut utiliser à la fois la fonction de comptage et de capture de position.

- Lorsque le compteur atteint sa valeur maxi, il repasse à 0 au prochain front (valeur maxi : 65535).

8-4-1- Configuration :

L'instruction SETUPCOUNTER permet de configurer le compteur.

Syntaxe : SETUPCOUNTER(<n° de compteur>,<Entrée>,<Filtre>)

<n° de compteur> : 0 ou 1

<Entrée> : Numéro de l'entrée (1 à 16)

<Filtre> : Validation du filtre : 0 pour sans filtrage, 1 pour avec filtrage.

Si le filtre n'est pas activé, la fréquence maxi est de 781Hz soit 1,24 ms sinon il dépend du paramètre Filtrage dans **Paramètres** / **Entrées Sorties Digitales** .

8-4-2- Ecriture :

L'instruction COUNTER(1 ou 2) permet d'initialiser le compteur à une valeur.

Syntaxe : COUNTER(<n° de compteur>) = <Val>

<n° de compteur> : Numéro de compteur (1 ou 2)

<Val> : Valeur comprise entre 0 et 65535

8-4-3- Lecture :

L'instruction COUNTER_S permet de lire le compteur.

Syntaxe : <Variable>=COUNTER_S(<n° de compteur>)

<Variable> : entier compris entre 0 et 65535

<n° de compteur>: Numéro de compteur (1 ou 2)

8-5- Boîte à cames

8-5-1- Introduction

Les boîtes à cames permettent de piloter des sorties logiques suivant des positions angulaires, linéaires par des instructions optimisées.

DPL accepte 2 boîtes à cames avec jusqu'à 4 segments par boîte. Par exemple, les sorties 3, 4 et 12 peuvent être affectées à la boîte et les autres sorties peuvent être utilisées ailleurs.

Les sorties d'une boîte sont remises à jour toutes les 1,3ms.

Les fonctions disponibles sont : CAMBOX, CAMBOXSEG, STARTCAMBOX et STOPCAMBOX .

Lors de la déclaration d'un segment, la valeur de début peut-être supérieure à celle de fin. Le zéro programme est pris en compte à chaque définition de segment.

Avant de déclarer une boite à cames vous devez passer l'axe en mode esclave avec GEARBOX(1,1)

8-5-2- Boîte à cames

Le variateur gère jusqu'à deux boites à cames de quatre segments chacune.

La source est soit la position du codeur moteur, soit la position du codeur maître (connecteur X2).

Dans le cas où la source est le moteur, les valeurs de début et de fin de segment sont directement liées à l'unité et à la mise à l'échelle paramétrées dans l'écran **Motion control** / **Configuration** / **Unités**.

Dans le cas où la source est le codeur maître :

- S'assurer que le nombre d'incréments par tour codeur maître a bien été entré dans la fenêtre **Paramètres** / **Codeur** : fonction = entrée codeur et résolution = 4000 pour un codeur 1000 par exemple.

- S'assurer dans le cas d'un axe infini que dans le menu **Motion control** / **Configuration** / **Maître** : module = activé et valeur = à rentrer dans l'unité et avec la mise à l'échelle du moteur esclave.

Avant de déclarer une boite à cames vous devez passer l'axe en mode esclave avec GEARBOX(1,1)

Un Gearbox avec des valeurs différentes de 1,1 agit sur la mise à l'échelle de l'axe maître.

Exemple : On souhaite un module maître égale à 15 tours codeur.

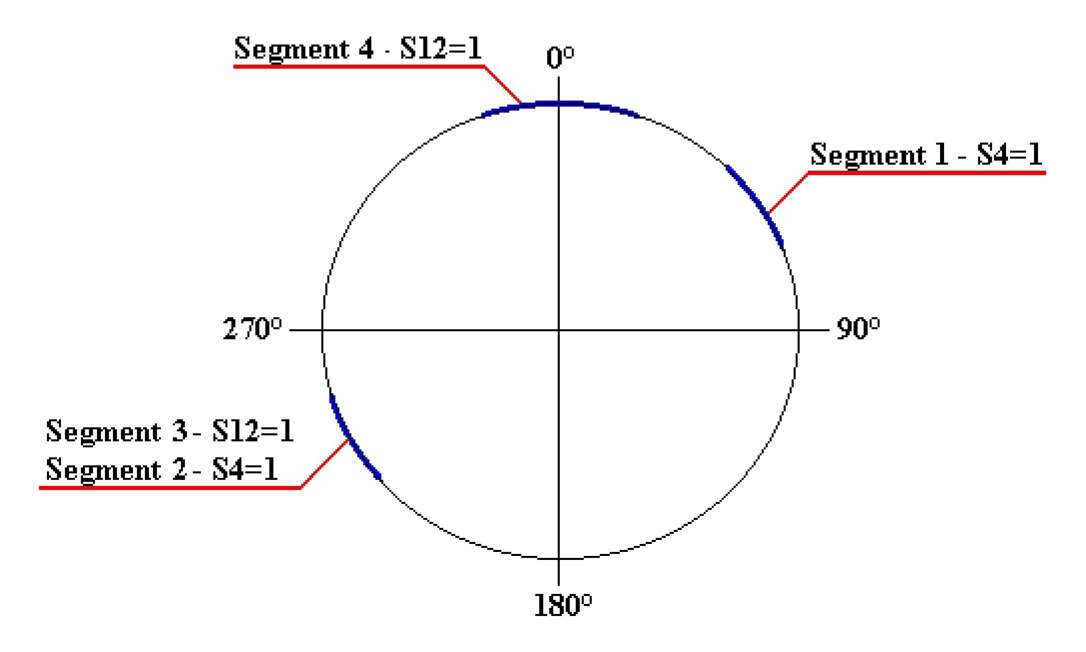
Codeur maître : 4000 incréments par tour.

Moteur esclave : Rin = 10, Rout = 1, Distance = 360° (voir écran **Motion control** / **Configuration** / **Unités**).

Un tour moteur esclave représente donc 36°.

Comme on a de façon interne une correspondance directe entre 1 tour codeur maître et 1 tour moteur, un tour codeur maître = 36°.

Dans l'instruction CAMBOXSEG, les début et fin de segments devront être compris entre 0° et 359.9°.



Dans cet exemple, le codeur maître est modulo 360. La boîte à cames s'écrit de la façon suivante :

GEARBOX (1,1) 'Passe l'axe en esclave

GEARBOXRATION (4096)

CAMBOX (1,1,4) 'La boîte à cames n°1 à 4 segments, source codeur maître

CAMBOXSEG(1,1,4,40,60) 'Le segment 1 de la boîte n°1 met la sortie 4 à 1 entre 40° et 60°

CAMBOXSEG(1,2,4,230,250) 'Le segment 2 de la boîte n°1 met la sortie 4 à 1 entre 230° et 250°

CAMBOXSEG(1,3,12,230,250) 'Le segment 3 de la boîte n°1 met la sortie 12 à 1 entre 200° et 400°

CAMBOXSEG(1,4,12,350,10) 'Le segment 4 de la boîte n°1 met la sortie 12 à 1 entre 350° et 10°

STATCAMBOX(1) ‘Démarrage de la boîte n°1

…

STOPCAMBOX (1) 'Arrêt de la boîte n°1

9- Liste des opérateurs et instructions

Pour connaître le temps d’exécution de chaque instruction, consulter le fichier DPL TIME INSTRUCTION.XLS dans le répertoire \Data du CD.

9-1- Programme

CALL	Appel de Sous-programme
NEXTTASK	Basculement immédiat à la tâche suivante
GOTO	Saut à une étiquette
PROG ... END PROG	Début d'un programme
SUB ... END SUB	Sous-programme
EXIT SUB	Sortie d'un sous-programme

9-2- Arithmétique

+	Addition
-	Soustraction
*	Multiplication
/	Division

9-3- Mathématique

FRAC	Partie fractionnelle
INT	Partie entière
MOD	Modulo

9-4- Logique

<<	Décalage à gauche
>>	Décalage à droite
AND	Opérateur ET

NOT	Opérateur complément
OR	Opérateur OU
XOR	Opérateur OU exclusif

9-5- Test

<	Inférieur
<=	Inférieur ou égal
<>	Différent
=	Egal / affectation
>	Supérieur
>=	Supérieur ou égal
IF	Test conditionnel

9-6- Contrôle de mouvement

- **Contrôle de l'axe :**

ACC	Accélération
ACC%	Accélération en pourcentage
AXIS	Contrôle la boucle d'asservissement
AXIS_S	Lit l'état de la boucle d'asservissement
BUFMOV_S	Nombre d'ordres en attente
CLEAR	Met à zéro la position de l'axe
CLEARMASTER	Met à zéro la position de l’axe maître
DEC	Décélération
DEC%	Décélération en pourcentage
FE_S	Erreur de poursuite
FEMAX_S	Limite d'erreur de poursuite
HOME	Prise d'origine
HOME_S	Etat de la prise d'origine

HOMEMASTER	Prise d'origine sur l'entré codeur
LOOP	Mode virtuel
MERGE	Définit l'enchaînement
MOVE_S	Etat du mouvement
ORDER	Numéro d'ordre du mouvement
ORDER_S	Numéro d'ordre courant
POS	Position à atteindre
POS_S	Position réelle
VEL	Vitesse
VEL%	Vitesse en pourcentage

- **Positionnement :**

MOVA	Mouvement absolu
MOVR	Mouvement relatif
SSTOP	Arrêt d'un axe sans attente
STOP	Arrêt d'un axe
STTA	Lance un mouvement absolu
STTI	Lance un mouvement infini
STTR	Lance un mouvement relatif

- **Synchronisation :**

GEARBOX	Arbre électrique
GEARBOXRATIO	Modifie le rapport de réduction d'un arbre électrique
STARTGEARBOX	Lance l'arbre électrique
STOPGEARBOX	Arrête l'arbre électrique

- **Capture :**

CAPTURE1 et CAPTURE2	Lancement de capture de position
REGPOS1_S et REGPOS2_S	Position capturée
REG1_S et REG2_S	Etat de la capture

9-7- Automate

- **Entrées / sorties TOR**

CAMBOX	Boîte à cames
CAMBOXSEG	Segment de boîte à cames
INP	Lecture d'une entrée logique
INPB	Lecture d'un bloc de 8 entrées
INPW	Lecture d'un bloc de 16 entrées
OUT	Ecriture d'une sortie
OUTB	Ecriture d'un bloc de 8 sorties
STARTCAMBOX	Lance une boîte à cames
STOPCAMBOX	Arrête une boîte à cames
WAIT	Attente d'une condition

- **Entrées / sorties analogiques**

ADC(1)	Entrée analogique n°1
ADC(2)	Entrée analogique n°2
DAC	Sortie analogique

- **Temporisations**

DELAY	Attente passive
LOADTIMER	Charge une temporisation dans une variable
TIME	Base de temps
TIMER	Compare une variable à Time

- **Compteurs**

COUNTER	Initialise un compteur à une valeur
SETUPCOUNTER	Configuration du compteur

COUNTER_S	Renvoie la valeur d'un compteur

9-8- Gestion des tâches

CONTINUE	Continue l'exécution d'une tâche
HALT	Arrête une tâche
RUN	Lance une tâche
SUSPEND	Suspend une tâche
STATUS	Etat d'une tâche

9-9- Flash, Sécurité, Divers

DISPLAY	Afficheur 7 segments
LOADPARAM	Permet de recharger les paramètres du variateur
LOADVARIABLE	Permet de charger les variables sauvegardées
RESTART	Redémarrage du variateur
SAVEPARAM	Permet de sauvegarder les paramètres du variateur
SAVEVARIABLE	Permet de sauvegarder les variables VR0..VR63, VL0..VL63
SECURITY	Définit les actions de sécurité
VERSION	Renvoie la version de l'Operating System

9-10- Liste aplhabétique

9-10-1- Addition (+)

Syntaxe : <Expression1> + <Expression2>

Types acceptés : Octet, Entier, Entier long et réel

Description : Cet opérateur additionne deux expressions et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides.
<Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1+VL2 'Résultat : VL3=15

Voir aussi : `-`, `*' et `/'.

9-10-2- Soustraction (-)

Syntaxe : <Expression1> - <Expression2>

Types acceptés : Octet, Entier, Entier long ou réel

Description : Cet opérateur soustrait l'<Expression2> de l'<Expression1> et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1-VL2 'Résultat : VL3=5

Voir aussi : '+', `*' et `/'.

9-10-3- Multiplication (*)

Syntaxe : <Expression1> * <Expression2>

Types acceptés : Octet, Entier, Entier long ou réel

Description : Cet opérateur multiplie l'<Expression1> par l'<Expression2> et retourne une valeur du même type que ces opérandes.

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

VL2=5

VL3=VL1*VL2 'Résultat : VL=50

Voir aussi : '+', '-' et `/'.

9-10-4- Division (/)

Syntaxe : <Expression1> / <Expression2>

Types acceptés : Octet, Entier, Entier long ou réel

Description : Cet opérateur divise l'<Expression1> par l'<Expression2>

Remarques : <Expression1> et <Expression2> doivent être des expressions numériques valides. <Expression1> et <Expression2> doivent être de même type. <Expression2> doit être différente de zéro. Cet opérateur retourne toujours une valeur réelle.

Exemple :

```
VL1=10
VL2=5
VL3=VL1/VL2          'Résultat : VL3=2
```

Voir aussi : '+', '-', `*'.

9-10-5- Inférieur (<)

Syntaxe : <Expression1> < <Expression2>

Types acceptés : Bit, Octet, Entier, Entier long ou réel

Description : Cet opérateur teste si <Expression1> est inférieure à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple :

```
VL1=10
IF VL1 < VL 2 ...
```

Voir aussi : '=', '>', '>=', '<=', '<>'.

9-10-6- Inférieur ou égal (<=)

Syntaxe : <Expression1> <= <Expression2>

Types acceptés : Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1>est inférieure ou égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple :

```
VL1 =10
IF VL1<= VL1 ...
```

Voir aussi : '=', '>', '>=', '<', '<>'.

9-10-7- Décalage à gauche (<<)

Syntaxe : <Expression1> << <Expression2>

Types acceptés : Octet ou Entier

Description : Cet opérateur déplace <Expression2> bits de <Expression1> de droite à gauche.

Remarques : <Expression2> représente le nombre de bits à déplacer. Le décalage n'est pas circulaire.

Exemple : VL1 = 4

VL2= VL1 << 2 'Résultat VL2= 16

Voir aussi : ' >> '.

Attention : Laisser un espace avant et après le décalage.

9-10-8- Différent (<>)

Syntaxe : <Expression1> <> <Expression2>

Types acceptés : Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> et <Expression2> sont différentes.

Remarques : <Expression1>et <Expression2> doivent être des expressions valides. <Expression1> et <Expression2> doivent être de même type.

Exemple : VL1=10

IF VL2<> VL1 ...

Voir aussi : '=', '>', '>=', '<', '<='

9-10-9- Affectation/Egalité (=)

Syntaxe : <Expression1> = <Expression2> Ou <Variable>=<Expression2>

Types acceptés : Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur affecte <Variable> à <Expression2> ou teste si <Expression1> est égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être des expressions valides. <Expression1>, <Expression2> et <Variable> doivent être de même type.

Exemple : VL1=1

```
BOUCLE:
VL1 = VL1 + 1
IF VL1 =10 GOTO SUITE
GOTO BOUCLE
SUITE :
```

Voir aussi : '>', '>=', '<', '<=', '<>'

9-10-10- Supérieur (>)

Syntaxe : <Expression1> > <Expression2>

Types acceptés : Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> est supérieure à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être de même type.

Exemple : IF VL1 > VL2 ...

Voir aussi : '=', '>=', '<', '<=', '<>'

9-10-11- Supérieur ou égal (>=)

Syntaxe : <Expression1> >= <Expression2>

Types acceptés : Bit, Octet, Entier, Entier long, réel

Description : Cet opérateur teste si <Expression1> est supérieure ou égale à <Expression2>.

Remarques : <Expression1> et <Expression2> doivent être de même type.

Exemple : IF VL1 >= VL2 ...

Voir aussi : '=', '>', '<', '<=', '<>'.

9-10-12- Décalage à droite (>>)

Syntaxe : <Expression1> >> <Expression2>

Types acceptés : Octet ou Entier

Description : Cet opérateur déplace <Expression2> bits de <Expression1> de gauche à droite.

Remarques : <Expression2> représente le nombre de bits à déplacer. Le décalage n'est pas circulaire.

Exemple : VL1 = 48

VL2 = VL1 >> 3 'Résultat VL2 = 12

Voir aussi : ' << '.

Attention : Laisser un espace avant et après le décalage.

9-10-13- ACC - Accélération

Syntaxe 1 : ACC = <Expression>

Syntaxe 2 : <Variable> = ACC

Unité : Expression, Variable : unité utilisateur par s^2 (Ex : mm/s^2, degré/s^2, tr/s^2…)

Types acceptés : <Expression> : réel

<Variable> : réel

Description : Cette instruction lit ou modifie l'accélération courante.

Remarques : <Expression> doit être une expression réelle valide. L'accélération courante peut être lue et modifiée à tout moment.

Exemple : ACC = 500

VR0 = 1000

ACC = VR0

Voir aussi : DEC, POS et VEL

9-10-14- ADC (1) – Entrée analogique 1

Syntaxe : <Variable>= ADC (1)

Unité : Variable : Volt

Limite : Variable : +/- 10V

Types acceptés : <Variable> : réel

Description : Cette fonction retourne la tension de l'entrée analogique n°1.

Exemple : VR1=ADC (1)

Voir aussi : DAC, ADC (2)

9-10-15- ADC (2) – Entrée analogique 2

Syntaxe : <Variable>= ADC (2)

Unité : Variable : Volt

Limite : Variable : +/- 10V

Types acceptés : <Variable> : réel

Description : Cette fonction retourne la tension de l'entrée analogique n°2.

Exemple : VR2 =ADC (2)

Voir aussi : DAC, ADC (1)

9-10-16- ACC% - Accélération en pourcentage

Syntaxe : ACC% = <Expression>

Limites : Expression : de 1 à 100

Types acceptés : <Expression> : Octet

Description : Cette fonction ajuste l'accélération courante en pourcentage du paramètre d'accélération.

Remarques : La valeur du paramètre accélération peut être entrée dans l'écran Motion control / Configuration / Profil de vitesse.

Exemple : ACC%=10 'L'accélération courante est de 10%

VB = 50

ACC%=VB0

Voir aussi : DEC%

9-10-17- AND – Opérateur ET

Syntaxe : <Expression1> AND <Expression2>

Types acceptés : Bit, Octet ou entier

Description : Cette fonction effectue un ET binaire entre deux expressions et retourne une valeur du type de l'opérande.

Remarques : <Expression1> et <Expression2> doivent être du même type.

Exemple : VB3=1001111b

VB4=1111110b

VB2=VB3 AND VB4 ‘VB2=1001110b

Voir aussi : OR, NOT, XOR et IF

9-10-18- AXIS – Contrôle la boucle d’asservissement

Syntaxe: AXIS ON | OFF

Description : Cette instruction est utilisée pour ouvrir et fermer la boucle d'asservissement.

Remarques : Lorsque l’axe est en boucle fermée (AXIS ON), toutes les instructions de mouvement sont transmises à l’axe par l’intermédiaire du buffer de mouvement et sont exécutées. Si l’axe passe en boucle ouverte (AXIS OFF), le buffer de mouvement est vidé, les instructions MOVE_S et FE_S retournent la valeur 0.

Exemple :

```
AXIS ON         'passage en boucle fermée
MOVA=1000       'déplacement à la position 1000
OUT (3) =1      'Sortie n°3 =1
MOVA=2000
OUT (3) =0
```

Attention : Voir les modes de déverrouillage du variateur dans le chapitre Logiciel DPL \ Menus et icônes \ Paramètres \ E/S digitales.

Voir aussi : AXIS_S, SECURITY

9-10-19- AXIS_S – Lit l’état de la boucle d’asservissement

Syntaxe : AXIS_S

Description : Cette fonction est utilisée pour lire l'état de la boucle d'asservissement et retourne une réponse 1 ou 0.

Remarques : Cette fonction peut être utilisée à tout moment pour voir si l'axe est asservi.

Exemple :

```
MOVA=100
If AXIS_S = 0 GOTO Error     'Erreur car l’axe est passé en ‘boucle
ouverte
```

Voir aussi : AXIS

9-10-20- BUFMOV_S

Syntaxe: <Variable>=BUFMOV_S

Types acceptés : <Variable> : Octet

Description : Cette fonction retourne le nombre de mouvements en attente dans le buffer du variateur. Le mouvement en cours d'exécution n'est pas comptabilisé par cette fonction.

Remarques : Cette fonction peut être utilisée après avoir lancé des mouvements, pour savoir si un mouvement est fini. Dans le cas où le buffer de mouvement se trouve plein, la tâche se bloque jusqu'à ce qu'une place soit libérée.

Exemple :

```
STTR=100
STTR=50
STTR=50
WAIT BUFMOV_S<2      'Attendre la fin du premier mouvement
```

9-10-21- CALL – Appel d'un sous-programme

Syntaxe : CALL <Nom>

Description : Cette instruction est utilisée pour appeler un sous-programme défini par un bloc SUB. <Nom> est le nom du bloc du sous-programme.

Remarques : Un sous-programme ne peut pas s'appeler. L'exécution de cette instruction provoque un basculement de tâche.

Exemple : CALL Mouvement

Voir aussi : SUB

9-10-22- CAMBOX – Boîte à cames

Syntaxe : CAMBOX (<Num boîte>, <Source>, <Nb Seg>)

Limites : Num boîte : de 1 à 2

Source : 0 si moteur ou 1 si codeur maître.

Nb seg : de 1 à 4

Types acceptés : Num boîte : Octet

Nb Seg : Octet

Description : Cette fonction définit une boîte à cames. Tout segment (CAMSEG) précédemment défini sur cette boîte est effacé.

Remarques : <Num boite> désigne un numéro de boîte

<Nb seg> est le nombre de segment dans la boîte. Si cette valeur est nulle la boite à came est détruite et doit être redéfinie si on veut la réutiliser.

Exemple : CAMBOX (1, 1,4) 'Boite à came n°1 de 4 segments dont la source est un codeur maître

Voir aussi : CAMBOXSEG

9-10-23- CAMBOXSEG – Segment de boîte à cames

Syntaxe : CAMBOXSEG (<Num boîte>, < Num Seg >, <Num sortie>, <Début>, <Fin>)

Limites : Num boîte : valeur de 1 à 2

Num Seg : valeur de 1 à 4

Num sortie : valeur de 1 à 10

Unité : Début, Fin : valeur dans l'unité utilisateur moteur

Types acceptés : Num boîte, Num seg, Num sortie : Octet

Début, Fin : réel

Description : Cette fonction définit un segment d'une boîte à came.

Remarques : <Num boîte> désigne la boîte. <Num seg> désigne le segment. <Num sortie> est la sortie à modifier. La sortie sera mise à 1 entre <Début> et <Fin>.

Exemple : CAMBOXSEG (1, 2, 4, 0,90) 'Le second segment de la boîte 1 met la 4ème sortie à 1 entre 0 et 90° (l'unité utilisateur définie étant le degré).

Voir aussi : CAMBOX

9-10-24- CAPTURE1 et CAPTURE2 - Lancement de capture de position

Syntaxe : CAPTURE1 (<Source>, <N° de l'entrée>, < Fenêtre >, <Mini>, <Maxi>, <Intérieur>)

Description : Les instructions CAPTURE1 et CAPTURE 2 sont utilisées pour enregistrer la position courante de l'axe ou d'un codeur maître.

Avec cette instruction, le variateur attend un front montant sur l'entrée capture. Quand le front est détecté, la position est stockée dans la variable REGPOS1_S. Le flag REG1_S est alors positionné à vrai.

Type accepté : <Source> 0 pour codeur moteur, 1 pour entrée codeur maître.

<N° de l’entrée> numéro l’entrée sur laquelle on attend le front montant (valeur de 1 à 16).

<Fenêtre> est vraie alors l'entrée n'est testée que lorsque l’axe est entre les positions <Mini> et <Maxi>.

<Interieur> permet de définir si le test s'effectue à l'intérieur ou à l'extérieur des bornes <Mini> et <Maxi>

<Mini> doit toujours être inférieur à <Maxi>.

Exemple : CAPTURE1 (0, 4, 1, 10, 20,1) 'Capture position du codeur moteur sur front montant de l’entrée 4 lorsque l’axe est entre 10 et 20.

WAIT REG1_S = 1 ‘Attente d’une capture

VR1 = REGPOS1_S ‘VR1 = valeur de la position lors de la capture

Voir aussi : REG1_S ou REG2_S, REGPOS1_S ou REGPOS2_S

9-10-25- CLEAR – Met à zéro la position de l’axe

Syntaxe : CLEAR

Description : Cette instruction met à zéro la position de l’axe.

Exemple : CLEAR

VR1=POS_S 'Résultat : VR1=0.0

9-10-26- CLEARMASTER - met à zéro la position du codeur maître

Syntaxe : CLEARMASTER

Description : Cette instruction met à zéro la position du codeur maître.

Exemple : CLEARMASTER

9-10-27- CONTINUE – Continue l’exécution d’une tâche

Syntaxe : CONTINUE <n° tâche>

Description : Cette instruction est utilisée pour continuer l'exécution d'une tâche supendue.

Remarques : < n° tâche > doit être le numéro d'une tâche suspendue. Cette fonction n'a pas d'effet sur une tâche stoppée ou en cours d'exécution.

Exemple tâche 1: Wait Inp(9)

RUN 2

Begin:

Wait Inp(9)

SUSPEND 2

Wait Inp(8)

CONTINUE 2

Goto Begin

Voir aussi : RUN, HALT, SUSPEND

9-10-28- COUNTER - Initialise le compteur à une valeur

Syntaxe : COUNTER (1 ou 2) = <Val>

Types acceptés : <Val> : valeur comprise entre 0 et 65535

Description : L'instruction COUNTER (1 ou 2) permet d'écrire une valeur dans les compteurs 1 ou 2.

Exemple : COUNTER (2)=VL1+1000

Voir aussi : SETUPCOUNTER

9-10-29- COUNTER_S – Renvoie la valeur d'un compteur

Syntaxe : <Variable>=COUNTER_S (<n° de compteur>)

Description : L'instruction COUNTER_S permet de lire le compteur.

Type acceptés : <Variable> entier compris entre 0 et 65535

<N° de compteur> numéro de compteur (1 ou 2)

Exemple : VI0 = COUNTER (1)

9-10-30- DAC - Sortie analogique

Syntaxe : DAC = <Expression>

Unité : Expression : Volts

Limites : Expression : de -10 à +10

Types acceptés : Expression : réel

Description : Cette fonction envoie une tension sur la sortie analogique du variateur.

Remarques : La sortie analogique peut également être lue.

Exemple : DAC=5.2

IF ADC (1)>DAC ...

Voir aussi : ADC (1), ADC (2)

9-10-31- DEC - Décélération

Syntaxe 1 : DEC = <Expression>

Syntaxe 2 : <Variable> = DEC

Unité : unité utilisateur par s^2 (Ex : mm/s^2, degré/s^2, tr/s^2…)

Types acceptés : Variable, Expression : réel

Description : Cette instruction lit ou modifie la décélération courante en unités par s^2.

Remarques : La décélération courante peut être lue et modifiée à tout moment.

Exemple : DEC = 500.

VR0 = 10000

DEC = VR0

Voir aussi : ACC, VEL

9-10-32- DEC% - Décélération en pourcentage

Syntaxe : DEC% = <Expression>

Limites : Expression de 0 à 100

Types acceptés : Expression : octet

Description : Cette fonction fixe la décélération courante en pourcentage du paramètre de décélération.

Remarques : La valeur du paramètre de décélération peut être entrée dans l'écran Motion control / Profil de vitesse.

Exemple : DEC% = 10 'Décélération courante 10 %

VB0 = 50

DEC% = VB0

Voir aussi : ACC% et VEL%

9-10-33- DELAY – Attente passive

Syntaxe : DELAY <Durée>

Unité : Durée : millisecondes

Types acceptés : Durée : Entier

Description : Cette fonction réalise une attente suivant la durée spécifiée. Elle bloque la tâche et provoque le basculement vers la tâche suivante.

Exemple : DELAY 500 'Délai de 0.5 s.

Où

VI12=500

DELAY VI12

9-10-34- DISPLAY – Afficheur 7 segments

Syntaxe : DISPLAY <Expression>

Types acceptés : Expression : Octet

Description : Cette Instruction fixe l'état de l'afficheur status 7 segments

Remarque : Chaque bit de <Expression> représente un segment. Le dernier bit n'étant pas utilisé.

Exemple : Display 109' Equivalent à Display 01101101b ou « 5 »

b1
b6 b7 b2
b5 b3
b4

9-10-35- EXIT SUB – Sortie d'un sous-programme

Syntaxe : EXIT SUB

Description : Cette instruction permet de sortir d'un sous-programme

Voir aussi : SUB

9-10-36- FEMAX_S – Limite d'erreur de poursuite

Syntaxe : FEMAX_S

Description : Ce flag est mis à 1 lorsque l'erreur de poursuite courante dépasse le seuil du paramètre erreur de poursuite accessible à partir du menu Paramètres / Sécurités / Position.

Remarques : Cette fonction est utilisée pour savoir si l'axe est passé en erreur de poursuite Il est nécessaire de traiter ce flag dans une tâche de gestion des défauts si l'instruction SECURITY (0) ou SECURITY (1) a été utilisée.

Remise à 0 du flag :

- Si l'entrée logique E1 est configurée en AUCUNE, FEMAX_S passe à 0 sur rencontre de l'instruction Axis On dans une tâche basic ou sur front montant du bouton enable à partir de la fenêtre principale du DPL.
- Si l'entrée logique E1 est configurée en VALIDATION, FEMAX_S passe à 0 sur front montant de cette entrée.
- Si l'entrée logique E1 est configurée en VALIDATION+DPL, FEMAX_S passe à 0 si l'entrée E1 = 1 et exécution dans une tâche basic de l'instruction Axis On.

Example: IF FEMAX_S = 1 GOTO Error

GOTO Debut

Error:

Voir aussi: FE_S, SECURITY

9-10-37- FE_S - Erreur de poursuite

Syntaxe : FE_S

Description : Cette fonction retourne une image de l'erreur de poursuite courante.

Remarques : Cette fonction est utilisée pour connaître la valeur courante de l'erreur de poursuite. On peut ainsi vérifier le comportement de la régulation en temps réel.

Exemple : VR1 = FE_S

Voir aussi : FEMAX_S

9-10-38- FRAC – Partie fractionnelle

Syntaxe : FRAC (<Expression>)

Types acceptés : <Expression> : réel

Description : Cette fonction restitue la partie fractionnelle de <Expression>.

Remarques : Cette fonction restitue une valeur réelle.

Exemple : VR2=3.0214

VR1=FRAC (VR2) 'Résultat VR2=0.0214

Voir aussi : INT

9-10-39- GEARBOX - Arbre électrique

Syntaxe : GEARBOX (<Numérateur>, <Dénominateur>)

Description : Cette instruction permet de réaliser un arbre électrique entre un codeur maître et le moteur (axe esclave).

Types acceptés: <Numérateur> Entier ou valeur de 0 à 8.

<Dénominateur> Entier ou valeur de 0 à 65535.

<Numérateur> / <Dénominateur> définit le rapport entre le codeur et l'axe esclave.

Remarques : Cette instruction est non bloquante pour la tâche (excepté si le buffer de mouvements est plein).Tant que la liaison entre le maître et l'esclave ne sera pas coupée, l'instruction MOVE_S sera égale à 1 (même si l'esclave est arrêté).

Exemples : GEARBOX (1, 2) ‘Rapport nominal : ratio 0.5

Voir aussi : GEARBOXRATIO, STARTGEARBOX, STOPGEARBOX

9-10-40- GEARBOXRATIO - Modifie le rapport de réduction d'un arbre électrique

Syntaxe : GEARBOXRATIO (<Ratio>)

Description : Cette instruction permet de modifier le rapport de réduction d'une liaison arbre électrique en cours de mouvement.

Types Acceptés :<Ratio> Entier ou valeur de 0 à 65535 : le rapport de l'arbre est défini par (<Ratio> × <Numérateur>) / (<Dénominateur> × 4096). <Numérateur> et <Dénominateur> sont les paramètres de l'instruction GEARBOX.

Remarques : L'instruction est non-bloquante et permet de changer de ratio sans arrêter l'arbre électrique.

Exemple : GEARBOXRATIO(2048)

Voir aussi : GEARBOX, STARTGEARBOX, STOPGEARBOX

9-10-41- GOTO – Saut à une étiquette

Syntaxe : GOTO <Label>

Description : Réalise un branchement à une étiquette

Remarques : Une étiquette est un nom suivi du caractère ":". L'exécution de cette instruction provoque le basculement vers la tâche suivante.

Exemple: GOTO Begin

...

Begin :

Voir aussi : IF

9-10-42- HALT – Arrêter une tâche

Syntaxe : HALT < n° tâche >

Description : Cette instruction est utilisée pour stopper une tâche en cours d'exécution ou suspendue.

Remarques : Cette fonction n'a pas d'effet sur une tâche déjà arrêtée. Elle n'affecte pas les mouvements en cours ni les buffers de mouvements.

Exemple:

```
Begin:
Wait Inp(8)=On
RUN 2
Wait Inp(8)=Off
HALT 2
Goto Begin
```

Attention Après la demande d'arrêt d'un tache, il est conseillé d'attendre que celle-ci soit terminée avec la commande Wait Status(n° de tâche).

Voir aussi : RUN, SUSPEND, CONTINUE

9-10-43- HOME – Prise d'origine

Syntaxe : HOME (<Type>, [Reference])

Description : Cette fonction force l'axe à se déplacer vers sa position d'origine en utilisant le <Type> de prise d'origine choisi. Cette instruction est bloquante pour la tâche tant que la prise d'origine n'est pas terminée et provoque le basculement vers la tâche suivante. La prise d'origine s'effectue à la vitesse programmée dans l'écran Motion control / Home.Les valeurs de <Type> sont :

0 : immédiate

1 : Sur Top Z (le variateur n'effectue aucun déplacement mais recalcule sa position par rapport au Top Z d'où une nouvelle position se situant entre +/- ½ tour moteur).

2 : Sur capteur sans dégagement en sens +

3 : Sur capteur avec dégagement en sens +

4 : Sur capteur sans dégagement en sens -

5 : Sur capteur avec dégagement en sens -

6 : Sur capteur et Top Z sans dégagement en sens +

7 : Sur capteur et Top Z avec dégagement en sens +

8 : Sur capteur et Top Z sans dégagement en sens -

9 : Sur capteur et Top Z avec dégagement en sens -

[Reference] de type réel, permet de donner une référence à la prise d'origine

Remarques : Utilisez AXIS Off pour arrêter une prise d'origine en cours. Si <Type> n'est pas spécifié, la valeur est celle indiquée dans l'écran Home du logiciel DPL.

Exemple : VR0=100

HOME (3, VR0) ‘Prise d'origine sur capteur avec dégagement en sens plus et ayant comme référence 100.

Nota : Si le paramètre référence n'est pas entré alors celui-ci est nul.

HOME (2) ‘est équivalent à VR0=0 puis HOME (2, VR0)

Voir aussi : HOME_S

9-10-44- HOME_S – Etat de la prise d'origine

Syntaxe : HOME_S

Description : Cette fonction indique l'état de la prise d'origine

Remarques : Cette fonction est utilisée pour savoir si la prise d'origine a été effectuée ou non. Pendant un cycle de prise d'origine, l'indicateur HOME_S est forcé en 0. Dès que le cycle est entièrement terminé, HOME_S passe à 1.

Exemple: IF HOME_S = OFF GOTO Suite

Suite :

Voir aussi : HOME

9-10-45- HOMEMASTER – Prise d'origine sur l'entrée codeur

Syntaxe : HOMEMASTER(<Type>)

Description : Cette fonction force permet de forcer à 0 la position interne du codeur maître. Les valeurs de <Type> sont :

0 : immédiate

1 : Sur Top Z

2 : Sur capteur de l'entré 5 (front montant)

Exemple : HOMEMASTER (0) 'Remet à 0 la position du maître.

Nota : Cette instruction n'est pas bloquante pour les tâches. Il est nécessaire de rajouter des tests dans les tâches DPL afin de savoir quand le HOMEMASTER est terminé.

9-10-46- IF - IF…

Syntaxe 1: IF <Condition> GOTO {<Etiquette>}

Description : Permet l'exécution conditionnelle, basée sur l'évaluation d'une expression.

Remarques : Le mot-clé IF commence une structure de contrôle IF.... Il doit apparaître avant toute autre partie de la structure. <Condition> doit être une expression booléenne.

Si <Condition> est vraie alors aller en <Etiquette>.

Example: IF VR1=150 GOTO SUITE

9-10-47- INP – Lecture d'une entrée TOR

Syntaxe : INP (<NuméroEntrée>)

Types acceptés : Numéro d'entrées de 1 à 16.

Description : Cette fonction donne l'état d'une entrée logique.

Remarques : <Entrée> doit représenter le numéro de l'entrée logique. Le type de donnée retournée est Bit.

Exemple : VF1 = INP (11)

Voir aussi : INPB, INPW, OUT, OUTB

9-10-48- INPB – Lecture d'un bloc 8 entrées

Syntaxe : INPB (<NuméroBlocEntrées>)

Types acceptés : Numéro d'entrées 1 ou 2.

Description : Cette fonction retourne l'état d'un bloc de 8 entrées logiques.

Remarques : <Entrées> doit représenter le numéro d'un bloc de 8 entrées. Le type donné retourné est octet.

Exemple : VB1=INPB (2)

Voir aussi : INP, INPW, OUT, OUTB

9-10-49- INPW – Lecture des 16 entrées logiques

Syntaxe : INPW

Description : Cette fonction donne l'état du bloc de 16 entrées logiques.

Remarques : Le type de données retourné est entier.

Exemple : VI2=INPW

Voir aussi : INP, INPB, OUT, OUTB

9-10-50- INT – Partie entière

Syntaxe : INT (<Variable>)

Types acceptés : Variable réel

Description : Cette fonction restitue la partie entière d'< Variable >.

Exemple :
```
VR1=25.36
VR2=INT (VR1)          'Résultat : VR2=25
```

Voir aussi : FRAC

9-10-51- LOADPARAM – Permet de recharger les paramètres du variateur

Syntaxe : LOADPARAM

Description : Permet de transférer dans la mémoire de travail RAM, les paramètres sauvegardés de la mémoire FLASH.

Voir aussi : SAVEPARAM

9-10-52- LOADVARIABLE - Permet de transférer les variables sauvegardés

Syntaxe : LOADVARIABLE

Description : Permet de transférer dans la mémoire de travail, les variables VR0 à VR63 et VL0 à VL63 sauvegardées de la mémoire FLASH.

Voir aussi : SAVEVARIABLE

9-10-53- LOADTIMER - Charge une temporisation dans une variable

Syntaxe : LOADTIMER (<VL n°XX>)=<Val>

Types acceptés : Val : entier long ou valeur numérique

Description : L'instruction LOADTIMER peut être utilisée pour établir des attentes actives. Elle stocke dans la variable VLXX, la somme de **Time** + <Val>

Remarques : Il est possible d'utiliser simultanément 256 timers.

Exemple : LOADTIMER (VL129)=4688 'Charge une temporisation de 3000ms dans la variable VL129

Voir aussi : TIMER

9-10-54- LOOP – Mode virtuel

Syntaxe: LOOP ON/OFF

Description : Cette fonction passe l'axe en mode virtuel et permet de tester un programme sans codeur ni moteur. Dans ce mode, ne pas brancher la puissance sur le connecteur X10

9-10-55- MERGE – définit l'enchaînement

Syntaxe : MERGE ON | OFF

Description : Cette instruction est utilisée pour activer ou désactiver l'enchaînement des mouvements consécutifs.

Exemple :

```
MERGE ON
TRAJA (1000,500)        'Mouvements enchaînés sans
TRAJA (1500,200)        'passage par une vitesse nulle
MERGE OFF
TRAJA (1800,700)        'passage par une vitesse nulle à la position 1500
```

9-10-56- MOD - Modulo

Syntaxe : <Expression1> MOD <Expression2>

Types acceptés : Expression1, Expression2 : Octet, Entier, Entier long

Description : Cet opérateur restitue le reste d'une division entière.

Exemple : VI10=5

VI10=VI10 MOD 2 'Résultat : VI10=1

9-10-57- MOVA – Mouvement absolu

Syntaxe : MOVA = <Distance>

Unité : Distance : unité utilisateur (Ex : mm, degré,…)

Types acceptés : Distance : réel ou valeur de type réel

Description : Déplace l'axe à une position absolue. L’exécution de l’instruction provoque le basculement vers la tâche suivante.

Remarques : La tâche attend la fin du mouvement (condition MOVE_S=0) avant d'exécuter la prochaine instruction. L’axe utilise les valeurs courantes de vitesse, d'accélération et de décélération.

Exemple : VR0 = 12000

MOVA = VR0 ou MOVA = 12000

Voir aussi : MOVR, STTA, STTR, STTI et MOVE_S

9-10-58- MOVE_S – Etat du mouvement

Syntaxe : MOVE_S

Types acceptés : Bit

Description : Cette fonction indique si l'axe est en mouvement.

Remarques : Si l’axe est en mode non asservi (AXIS OFF), l’instruction MOVE_S = 0. Si l’axe est en mode asservi, MOVE_S est égale à 0 si les 4 points suivants sont vrais :

Le mouvement courant est terminé (trajectoire théorique terminée).

L’erreur de poursuite est à l’intérieur de la fenêtre de positionnement (+/- la valeur entrée dans le menu Paramètres / Sécurités / Position).

Le buffer de mouvement est vide.

Dans le cas d’un axe esclave lié par une fonction GEARBOX, le lien doit avoir été coupé.

Si l’un de ces points est faux, l’instruction MOVE_S retourne la valeur 1.

Exemple : STTA = VR10

WAIT MOVE_S = OFF ' Attente que l'axe soit arrêté

9-10-59- MOVR – Mouvement relatif

Syntaxe : MOVR = <Distance>

Types acceptés : Distance : réel

Description : Déplace l'axe à une position relative. L’exécution de l’instruction provoque le basculement vers la tâche suivante.

Remarques : La tâche attend la fin du mouvement (condition MOVE_S=0) avant d'exécuter la prochaine instruction. L’axe utilise les valeurs courantes de vitesse, d'accélération et de décélération.

Exemple: VR1 = 1200

MOVR = VR1 ou MOVR = 1200

Voir aussi: MOVA, STTA, STTR, STTI, MOVE_S

9-10-60- NEXTTASK

Syntaxe: NEXTTASK

Description : Instruction permettant de faire un basculement immédiat vers la tâche suivante.

9-10-61- NOT – Opérateur complément

Syntaxe : NOT (<Expression>)

Types acceptés : Expression : Bit, Octet, Entier

Description : La fonction NOT retourne le complément.

Exemple : VB1=15

VB2=NOT VB1 'Résultat VI2=140

Voir aussi : AND, OR, XOR

9-10-62- OR - Opérateur ou

Syntaxe : <Expression1> OR <Expression2>

Types acceptés : Expression1, Expression2 : Bit, Octet, Entier

Description : Cette fonction effectue un OU binaire entre deux expressions.

Remarques : <Expression1> et <Expression2> doivent être du même type. Cette fonction restitue le même type de donnée que ses arguments

Exemple : VI12=VI12 OR 000FFh

Voir aussi : AND, NOT, XOR et IF

9-10-63- ORDER – Numéro d’ordre du mouvement

Syntaxe 1 : ORDER = <Valeur>

Syntaxe 2 : ORDER

Types acceptés : Valeur : entre 0 et 65535

Description : Cette instruction fixe le numéro d'ordre+1 du prochain mouvement ou lit le numéro d'ordre du dernier mouvement déposé.

Remarques : Cette instruction peut être utilisée avec la fonction ORDER_S.

Exemple:

```
ORDER = 0
STTA = 50
VB1 = ORDER          'Résultat : VB1=1
```

Voir aussi : ORDER_S

9-10-64- ORDER_S – Numéro d’ordre courant

Syntaxe: ORDER_S

Types acceptés : Entier

Description : Cette fonction retourne le numéro du mouvement en cours d'exécution.

Remarques : Cette fonction peut être utilisée pour connaître l'état du mouvement.

Exemple:

```
ORDER=0
STTA = 50
STTA = 100
STTA = 50
```

IF ORDER_S=2 ...'Le second mouvement est commencé.

Voir aussi : ORDER

9-10-65- OUT – Ecriture d'une sortie

Syntaxe : OUT (<NumSortie>) = <Expression>

Types acceptés : Expression : Bit

Description : Cette fonction envoie un état logique à une sortie TOR.

Remarques : <Sortie> doit représenter le numéro d'une sortie de 1 à 10

Exemple : OUT (10) = ON

Voir aussi : INP, INPB, INPW, OUTB

9-10-66- OUTB – Ecriture d'un bloc de 8 sorties

Syntaxe : OUTB (<NuméroBlocSorties>) = <Expression>

Types acceptés : <Expression> : de type octet

(<NuméroBlocSorties> : 1 ou 2

Description : Cette fonction envoie des états logiques à un bloc de 8 sorties TOR.

Exemple : OUTB (1)=15

Voir aussi : INP, INPB, INPW, OUT

9-10-67- POS – Position à atteindre

Syntaxe 1 : POS = <Expression>

Syntaxe 2 : POS

Types acceptés : Expression : réel

Description : Cette fonction retourne ou fixe la position à atteindre dans l'unité choisie.

Remarques : Cette fonction est utilisée pour changer la position finale en cours de mouvement. La position peut être modifiée à tout moment.

Exemple :

```
STTA = 5000            'Départ de l'axe
WAIT INP (10) = On     'Attente Cellule
```

POS = POS_S+50. 'Arrêt 50 mm après le capteur

WAIT MOVE_S = OFF 'Attente arrêt de l'axe

Voir aussi : ACC, DEC, VEL

9-10-68- POS_S – Position réelle

Syntaxe : <Expression> = POS_S

Types acceptés : Expression : réel

Description : Cette fonction retourne la position réelle de l'axe.

Remarques : On peut ainsi obtenir l'image en temps réel de la position de l'axe.

Exemple :

```
STTA = 100          'Départ de l'axe
OUT (5) = 1         'Activation sortie n°5
BOUCLE:
VR1=POS_S
IF VR1<50 GOTO BOUCLE
OUT (5) = 0         'Désactivation de la sortie n°5
```

Voir aussi : VEL_S

9-10-69- PROG ... END PROG – Début d'un programme

Syntaxe : PROG

Description : Ce mot-clé commence un bloc de programme principal. Il est également utilisé pour identifier la fin d'un bloc de programme principal lorsqu'il est précédé de END.

Remarques : Un et seulement un bloc PROG - END PROG doit être défini dans un programme

Exemple :

```
PROG
...
END PROG
```

9-10-70- READPARAM - Lecture d'un paramètre

Syntaxe: <Variable> = READPARAM (<Index>, <Sub-Index>)

Types acceptés : <Variable> entier long

<Index> valeur de 0 à 65535

<Sub-Index> valeur de 0 à 255

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : VL0 = READPARAM (8448,1) ‘Renvoie le numéro du défaut du variateur

9-10-71- REG1_S ou REG2_S - Etat de la capture

Syntaxe : <VFx>=REG1_S

Description : Cette fonction indique si une capture de position a été effectuée.

Remarques : La valeur retournée n'est vraie qu'une fois par capture. REG1_S est remis automatiquement à 0 sur une opération de lecture et lorsqu’il vaut 1. Sur une relance d’une autre capture et si REG1_S vaut 1 alors REG1_S est mis à 0.

Exemple :

```
CAPTURE1 (0, 4, 1, 10, 20,1)   'Capture de la position du codeur moteur
                               ‘Sur front montant de l’entrée 4
                               ‘Lorsque l’axe est entre 10 et 20.
WAIT REG1_S = 1                ‘Attente d’une capture
VR1 = REGPOS1_S                ‘VR1 = valeur de la position lors de la capture
```

Voir aussi : CAPTURE1 ou CAPTURE2, REGPOS1_S ou REGPOS2_S

9-10-72- REGPOS1_S ou REGPOS2_S - Position capturée

Syntaxe : <VR XX>=REGPOS1_S

Description : Cette fonction retourne la dernière position capturée sur l'axe par l’exécution de l’instruction CAPTURE1.

Exemple :

```
CAPTURE1 (0, 4, On, 10, 20, On) 'Capture de la position du codeur moteur
                               ‘Sur front montant de l’entrée 4
                               ‘Lorsque l’axe est entre 10 et 20.
WAIT REG1_S = ON               ‘Attente d’une capture
VR1 = REGPOS1_S                ‘VR1 = valeur de la position lors de la capture
```

Voir aussi : CAPTURE1 ou CAPTURE2, REG1_S ou REG2_S

9-10-73- RESTART – Redémarrage du système

Syntaxe : RESTART

Description : Redémarre le système de la même manière qu'une mise sous tension.

9-10-74- RUN – Lance une tâche

Syntaxe : RUN < n° tâche >

Description : Cette instruction est utilisée pour lancer une tâche à l’arrêt (ex : tâche déclarée en démarrage manuel).

Remarques : Cette fonction n'a pas d'effet sur une tâche suspendue ou déjà lancée.

Exemple:
```
Debut:
Wait Inp(11)=On
RUN 3
Wait Inp(11)=Off
HALT 3
Wait Status (3) =0
Goto Debut
```

Attention Après la demande d’arrêt d’une tâche, il est conseillé d’attendre que celle-ci soit terminée avec la commande Wait Status(n° de tâche).

Voir aussi : CONTINUE, HALT, SUSPEND

9-10-75- SAVEPARAM - Permet de sauvegarder les paramètres du variateur

Syntaxe : SAVEPARAM

Description : Les paramètres du variateur en RAM EXTERNE sont sauvegardés en mémoire XFLASH.

Remarque : La FLASH a une durée de vie de 5000 cycles d’écriture.

Voir aussi : LOADPARAM

Attention : Consulter notre service technique avant l’utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

9-10-76- SAVEVARIABLE – Permet de sauvegarder les variables

Syntaxe : SAVEVARIABLE

Description : Les variables en RAM VR0 à VR63, VL0 à VL63 sont sauvegardées en mémoire FLASH.

Le variateur passe automatiquement en AXIS OFF

Remarque : La FLASH a une durée de vie de 5000 cycles d'écriture.

Voir aussi : LOADVARIALBE

Attention : Consulter notre service technique avant l'utilisation de cette instruction sous peine de dégradation prématurée de la mémoire FLASH

9-10-77- SECURITY – Définit les actions de sécurités

Syntaxe : SECURITY (<Niveau>)

Description : Cette instruction est utilisée pour définir comment le système doit réagir si une erreur de poursuite sur l'axe est détectée <Niveau> détermine le niveau de sécurité. Les valeurs par défaut à la mise sous tension sont SECURITY (2).

Niveau	Err. 12 sur afficheur	Flag Femax = 1	Etat de l'instruction Axis_s	S1 (ready) = 0
0	Non	1	Axis_s = On	1
1	Non	1	Axis_s = Off	1
2	Oui	1	Axis_s = Off	0

Remarques : Si l'instruction SECURITY est utilisée, le niveau de sécurité peut être diminué suivant l'écriture du programme. Il est conseillé de ne pas utiliser cette instruction.

Exemple : SECURITY (0) ' L'axe reste asservi en cas d'erreur de poursuite

Nota : Le flag Femax_S est remis à 0 lorsque l'on repasse en mode asservi (Axis On).

9-10-78- SETUPCOUNTER – Configure un compteur

Syntaxe : SETUPCOUNTER (<1 ou 2>, <Num Entrées>, <Filtre>)

Types acceptés : <Filtre> : bit

Description : Cette instruction permet de configurer les compteurs 1 ou 2

Remarques : <Num Entrée> : Numéro de l'entrée de 1 à 16

<Filtre> : Validation du filtre : 0 pour sans filtrage, 1 pour avec filtrage.

Voir aussi : COUNTER

Attention : Si le filtre n'est pas activé, la fréquence maxi est de 781Hz soit 1,24 ms sinon il dépend du paramètre Filtrage dans **Paramètres** / **Entrées Sorties Digitales**.

9-10-79- SSTOP – Arrêt d'un axe

Syntaxe : SSTOP

Description : Cette fonction stoppe l'axe avec la décélération courante. La fonction n'est pas bloquante pour la tâche.

Remarques : Si l'axe est lié avec la fonction GEARBOX alors l'axe s'arrête.

L'instruction SSTOP vide le buffer de mouvement et stoppe l'axe en utilisant la décélération courante. Cette instruction n'est pas bloquante et n'attend pas que MOVE_S soit égal à 0.

Exemple : SSTOP

Voir aussi : STTA, STTR, STTI, GEARBOX, CAMBOX

9-10-80- STARTCAMBOX – Lance une boîte à cames

Syntaxe : STARTCAMBOX (<Num boîte>)

Description : Cette instruction lance une boîte à cames précédemment définie.

Remarques : Si la boîte à cames n'est pas définie, la fonction n'a pas d'effet.<Num boîte> est le numéro utilisé dans la fonction CAMBOX.

Exemple : GEARBOX (1,1) 'Passe l'axe en mode esclave

STARTCAMBOX (1)

Voir aussi : CAMBOX

9-10-81- STARTGEARBOX - Lance l'arbre électrique

Syntaxe : STARTGEARBOX (<Accélération>)

Description : Cette instruction permet de lancer un arbre électrique suivant une accélération et un rapport défini précédemment par GEARBOX et GEARBOXRATIO.

Types Acceptés : <Accélération> valeur de 0 à 65535 ou variable entier

Remarques : La phase d'accélération sera de : (Ratio × 640µs) / Accélération, avec Ratio défini par GEARBOXRATIO.

Exemples : STARTGEARBOX (512) 'Lance l'arbre électrique avec une phase

… 'D'accélération de Ratio×640µs/512

Voir aussi : GEARBOX, GEARBOXRATIO, STOPGEARBOX

9-10-82- STATUS – Etat d'une tâche

Syntaxe : STATUS (<n° tâche>)

Description : Cette fonction retourne l'état d'une tâche

Remarques : Les valeurs possibles sont :

0 : La tâche est stoppée

1 : La tâche est suspendue

2 : La tâche est en cours d'exécution

Exemple : Run 2

Wait Status(2)=0

9-10-83- STOP - Arrêt d'un axe

Syntaxe : STOP

Description : Cette fonction stoppe l'axe avec la décélération courante. La fonction est bloquante tant que l'axe n'est pas arrêté.

Remarques : Si l'axe est un axe lié avec la fonction GEARBOX, alors l'axe s'arrête.

L'instruction STOP vide le buffer de mouvement et stoppe l'axe en utilisant la décélération courante. Cette instruction est bloquante tant que MOVE_S est différent de 0.

Exemple : STOP

Voir aussi : STTA, STTR, STTI, GEARBOX

9-10-84- STOPCAMBOX – Arrête une boîte à cames

Syntaxe : STOPCAMBOX (<Num boîte>)

Description : Cette instruction arrête une boîte à cames précédemment définie.

Remarques : <Num boîte> est le numéro utilisé dans la fonction CAMBOX. Cette fonction ne détruit pas la boîte.

Exemple : STOPCAMBOX (1)

Voir aussi : CAMBOX, CAMBOXSEG, STARTCAMBOX

9-10-85- STOPGEARBOX - Arrête l'arbre électrique

Syntaxe : STARTGEARBOX (<Décélération>)

Description : Cette instruction permet d'arrêter un arbre électrique suivant une décélération et un rapport défini précédemment par GEARBOX.

Types acceptés : <Décélération> valeur de 0 à 65535 ou variable entier

Remarque : La phase de décélération sera est de : (Ratio × 640µs) / Décélération

Exemples :

```
STOPGEARBOX (256)        'Arrêt de l'arbre électrique avec une phase
                         'De décélération de Ratio×640µs/256
WAIT MOVE_S=0   'de décélération de  (3687 × 640 / 2)=1,18s
VR0=POS_S       'Stocke la position réel de l'axe dans VRO
HOME (0, VR0)   'Copie la position réel de l'axe dans le position théorique
```

Voir aussi : GEARBOX, GEARBOXRATIO, STARTGEARBOX

9-10-86- STTA – Lance un mouvement absolu

Syntaxe : STTA = <Distance>

Types acceptés : Distance : réel

Description : Lance un mouvement à une position absolue.

Remarques : Le système n'attend pas la fin du mouvement (MOVE_S=0) et exécute la prochaine instruction. L'axe utilise la vitesse, l'accélération et la décélération courante

Exemple :

```
VR0 = 1200
STTA = VR0 ou STTA = 1200
WAIT MOVE_S = OFF
```

Voir aussi : MOVA, MOVR, STTR, STTI

9-10-87- STTI – Lance un mouvement infini

Syntaxe : STTI (+ ou -)

Description : Lance un mouvement infini.

Remarques : Le système n'attend pas la fin du mouvement et exécute la prochaine instruction. Vous devez utiliser l'instruction STOP ou SSTOP pour arrêter le mouvement. L'axe utilise la vitesse et l'accélération courante.

Exemple :

```
STTI (+)        ' commence un mouvement infini
                ' dans la direction positive.
```

Voir aussi : MOVA, MOVR, STTA, STTR, STOP

9-10-88- STTR – Lance un mouvement relatif

Syntaxe : STTR = <Distance>

Types acceptés : Distance : réel

Description : Lance un mouvement relatif.

Remarques : Le système n'attend pas la fin d'un mouvement (MOVE_S=0) avant d'exécuter la prochaine instruction. L'axe utilise la vitesse, l'accélération et la décélération courante

Exemple : VR0 = 420

STTR = VR0 ou STTR = 420

Voir aussi : MOVA, MOVR, STTA, STTI

9-10-89- SUB … END SUB – Sous-programme

Syntaxe : SUB <Nom>

Description : Ce mot-clé commence un bloc de sous-programme et est aussi utilisé pour définir la fin d'un bloc de sous-programme quand il est précédé de END.

Remarques : Les blocs SUB - END SUB doivent être en dehors d'un bloc PROG – END PROG.

Exemple: SUB Move

...

END SUB

9-10-90- SUSPEND – Suspend une tâche

Syntaxe : SUSPEND < n° tâche >

Description : Cette instruction suspend une tâche en cours d'exécution.

Remarques : Cette instruction n'a pas d'effet sur les tâches stoppées. Les mouvements présents dans le buffer de mouvement continuent à s'exécuter.

Exemple : Wait Inp(12)

RUN 4

Begin:

Wait Inp(12)

SUSPEND 4

Wait Inp(12)

CONTINUE 4

Goto Begin

Voir aussi : RUN, CONTINUE, HALT

9-10-91- TIME - Base de temps étendue

Syntaxe : <VLx> = TIME

Description : La variable système TIME peut être utilisée pour établir des attentes actives. C'est un entier long qui représente le nombre de 064 millièmes de secondes écoulées depuis la dernière mise sous tension.

Exemple : VL2=TIME

VL2=VL2 + 7812 'Charge une temporisation de 5000ms

ATTENTE:

VL3=TIME

IF VL3<VL2 GOTO ATTENTE

Attention : La fonction TIME ne fonctionne pas dans un test

9-10-92- TIMER – Comparaison une variable à Time

Syntaxe : TIMER (<VL XX>)

Description : Cette instruction compare la variable système **Time** et le contenu de la variable VLXX :

TIMER (VLXX) =1 si Time<=VLXX (temporisation en cours).

TIMER (VLXX)=0 si Time>VLXX (temporisation écoulée).

Types acceptés : VL XX : Variable du type entier long

Exemple : LOADTIMER (VL122)=4688 'Charge un temporisation de 3s

WAIT (TIMER (VL122)=0) 'Attente temporisation écoulée

9-10-93- TRAJA – Trajectoire absolue

Syntaxe : TRAJA (<Position>, <Vitesse>)

Types acceptés : <Position> du type réel

<Vitesse> du type réel

Description : Cette instruction effectue une trajectoire complexe. L'exécution de cette tâche provoque le basculement vers la tâche suivante.

Remarques : L'axe utilise l'accélération et la décélération courante.

Exemple :

```
MERGE On                    'Passage en petite vitesse
TRAJA (1000.00, VR0)        'à la position 1000,
TRAJA (1500.00, VR1)        ' sans arrêt de l'axe
MERGE Off
```

Voir aussi : STTA, MERGE, TRAJR

9-10-94- TRAJR – Trajectoire relative

Syntaxe : TRAJR (<Position>, <Vitesse>)

Types acceptés : <Position> du type réel

<Vitesse> du type réel

Description : Cette instruction effectue une trajectoire complexe. L'exécution de cette tâche provoque le basculement vers la tâche suivante.

Remarques : L'axe utilise l'accélération et la décélération courante.

Exemple :

```
MERGE On                    'Passage en petite vitesse
TRAJR (200.00, VR0)
TRAJR (1000.00, VR0)        'à la position 1200,
TRAJR (1500.00, VR1)        ' sans arrêt de l'axe
MERGE Off
```

Voir aussi : STTR, MERGE, TRAJA

9-10-95- VEL - Vitesse

Syntaxe : VEL = <Expression>

Unité : Expression : unité utilisateur par seconde (Ex : mm/s, tr/s, degré/s).

Types acceptés : Expression : réel

Description : Cette valeur spécifie la vitesse courante en unité par seconde.

Remarques : <Expression> doit être une expression réelle valide. Cette valeur de vitesse peut être modifiée à tout moment.

Exemple : VEL = 2000

Voir aussi : ACC, DEC, POS

9-10-96- VEL%

Syntaxe : VEL% = <Expression>

Limite : Expression : de 0 à 100 ou <VB n°XX>

Types acceptés : Expression : octet

Description : Cette fonction ajuste la vitesse courante en pourcentage du paramètre de vitesse de l'écran Motion control / Configuration / Profil de vitesse.

Exemple : VB0 = 50

VEL% = VB0

Voir aussi : ACC%, DEC%

9-10-97- VERSION – Version de l'operating system (Firmware)

Syntaxe : <VI n°XX>=VERSION

Description : Cette fonction retourne la version de l'Operating System.

9-10-98- WAIT - Attente d'une condition

Syntaxe : WAIT <Condition>

Description : Cette instruction permet au système d'attendre que la condition soit vraie.

Exemple : WAIT INP (11)=On 'Attente passive

9-10-99- WRITEPARAM - Ecriture d'un paramètre

Syntaxe : READPARAM (<Index>, <Sub-Index>) = <Variable>

Types acceptés : <Variable> du type entier long

<Index> de 0 à 65535

<Sub-Index> de 0 à 255

Description : Cette fonction permet de lire via le bus CANopen, les paramètres du variateur.

Exemple : WRITEPARAM (9984,6) = 1 ‘Active le modulo sur l’axe

9-10-100- XOR – Opérateur ou exclusif

Syntaxe : <Expression1> XOR <Expression2>

Types acceptés : Expression1, Expression2 : Bit, Octet, Entier

Description : Cette fonction fait un Ou Exclusif entre les expressions.

Remarques : <Expression1> et <Expression2> doivent être du même type de donnée. Cette fonction restitue le type de donnée de <Expression1>.

Exemple: VB0 = VB0 XOR VB1

IF VB0 = 0 GOTO Error

Voir aussi: AND, OR, NOT, IF

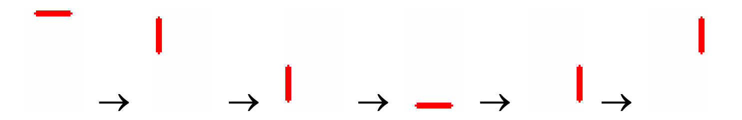
10- Annexes

10-1- Afficheur STATUS 7 segments

10-1-1- Description des messages :

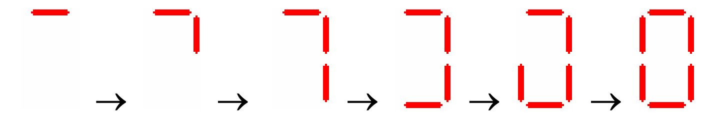
- **Phase d'initialisation du BOOT :**

Tous les segments de l'afficheur clignotent plusieurs fois puis s'allument dans l'ordre suivant :



- **Phases d'initialisation du système d'exploitation :**

Les segments s'allument dans l'ordre suivant mais pendant des temps différents :



L'initialisation complète du variateur dure 7s.

- **Après les initialisions :**

La sortie variateur prêt s'active (S1), si le DPL est utilisé : les tâches automatiques sont lancées et il ne doit rester qu'un point qui clignote.

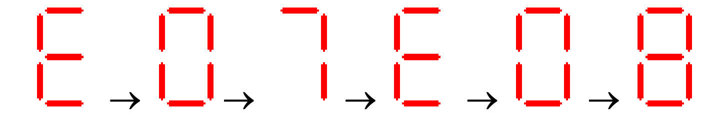
Si le DPL n'est pas utilisé, on fait tourner les segments en même temps et dans le même sens que le moteur.

Si le DPL est utilisé, on laisse uniquement le point. Les segments sont modifiés par l'exécution de l'instruction Display dans le programme DPL.

- **Pendant l'utilisation du variateur :**

Sur apparition d'une erreur : Les numéros des erreurs détectées sont affichés par ordre croissant puis rebouclage sur la première erreur.

Ex : Pour une erreur de température moteurE7 et une erreur de codeur E8, on aura :

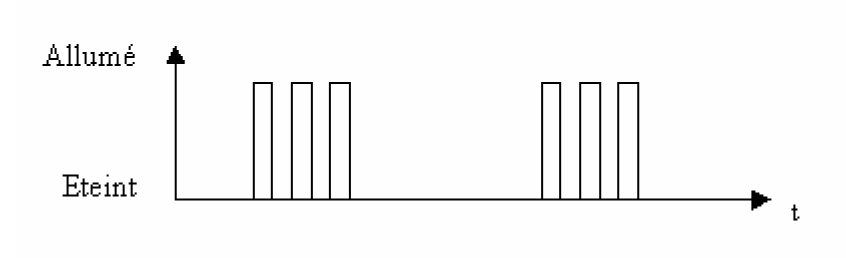


Sur disparition d'un défaut : Disparition du numéro d'erreur et retour à un affichage normal (comme après l'initialisation)

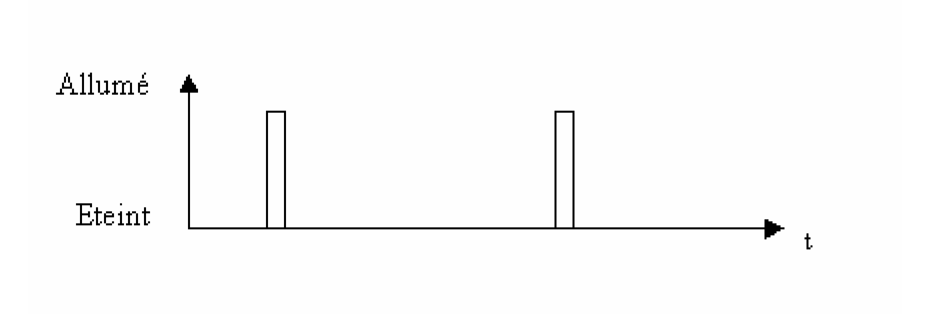


Clignotement du point :

Si liaison système en cours (signal RTS monté) :



Si pas de liaison système en cours :



10-1-2- Messages d'erreur :

- **Liste des erreurs :**

E01 Sur-tension DCBus : une surtension a été détectée sur le bus continu interne. Ce défaut peut être dû à une surtension sur le réseau ou à la résistance de ballast qui n'est pas suffisante.

E02 Sous-tension DCBus : une tension minimale a été détectée sur le bus continu interne. Ce défaut est géré uniquement lorsque le variateur est activé (Enable = ON).

E03 I^2t moteur : I^2t moteur détecté.

E04 Sur-courant : un courant supérieur au courant maximal a été détecté.

E05 Court-circuit : un court-circuit entre phases ou la mise à la terre d'une phase du moteur a été détecté.

E06 Température IGBT : température maximale atteinte dans le variateur.

E07 Température moteur : température maximale atteinte dans le moteur.

E08 Erreur resolveur : Signaux résolveur défectueux (Vérifiez le câble et les connecteurs résolveur du moteur).

E09 Paramètres invalides : erreur de checksum sur les paramètres du variateur.

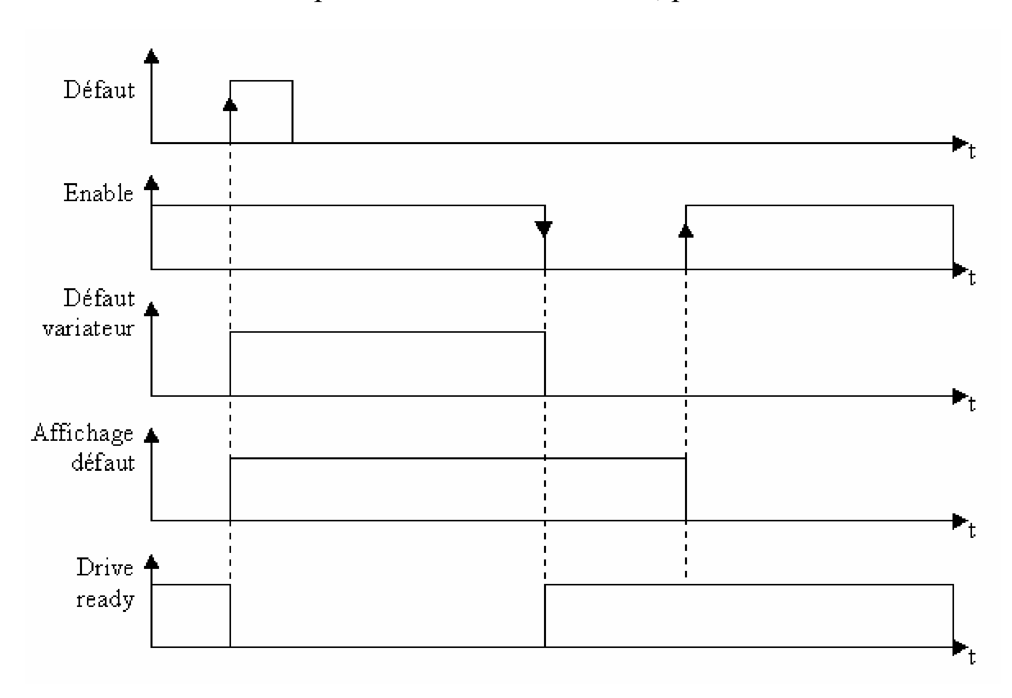
E 10 Défaut modèle de variateur : le fichier de paramètre ne correspond pas au modèle de variateur.

E 11 Erreur DPL : une erreur a été détectée pendant l'exécution des tâches DPL.

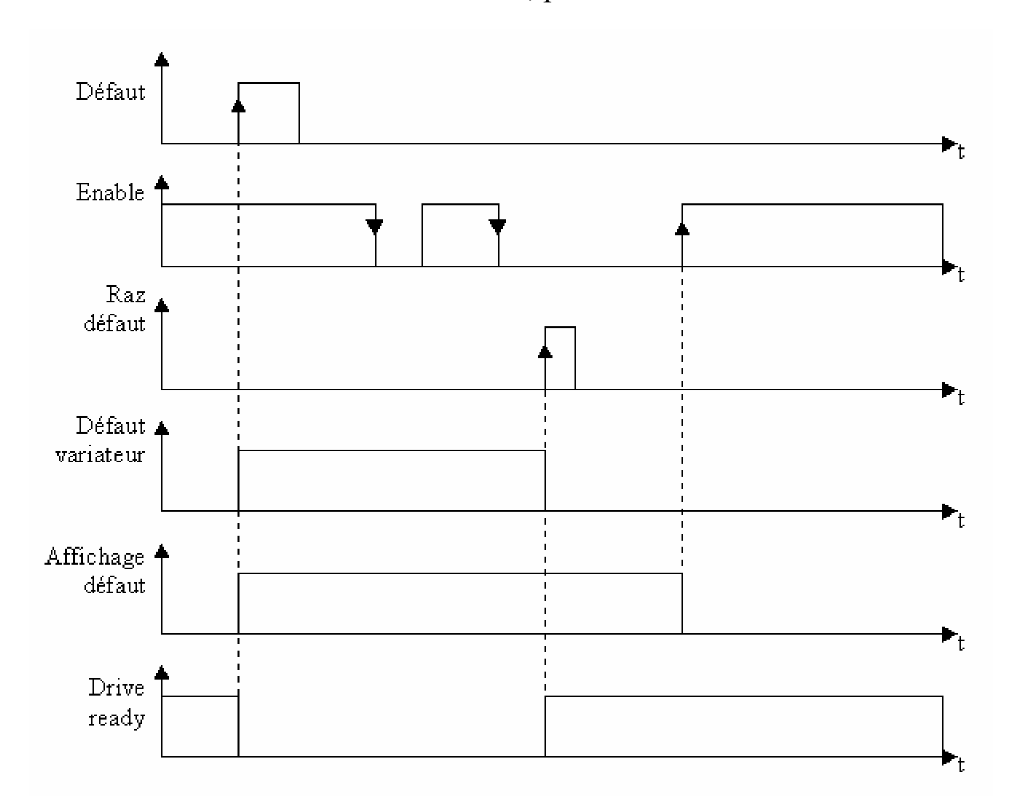
E 12 Erreur de poursuite : le variateur a dépassé l'erreur de poursuite.

- **Suppression des défauts :**

·Si l'entrée E4 n'est pas utilisée en RAZ défaut, procéder comme suivant :



·Si l'entrée E4 est utilisée en RAZ défaut, procéder comme suivant :



10-2- CANopen :

10-2-1- Définition :

- Introduction :

Le bus CAN (Controller Area Network) est apparu au milieu des années 80 pour répondre aux besoins de la transmission de données dans le secteur automobile. Ce type de bus permet d'obtenir des taux de transfert élevés.

Les spécifications du CAN définissent 3 couches parmi le modèle OSI : la couche physique, la couche liaison des données et la couche application. La couche physique définit le mode de transmission des données en fonction du support de transmission. La couche liaisons des données représente le noyau du protocole CAN puisque cette couche est responsable de la trame à envoyer, de l'arbitrage, de la détection des erreurs, etc… . La dernière couche est la couche application appelée aussi CAL (CAN Application Layer). Celle-ci est donc une description générale du langage pour les réseaux CAN qui offre de nombreux services de communication.

CANopen est un type de réseau qui est basé sur le système du bus série et de la couche application CAL. CANopen ne propose qu'une partie des services de communication offerte par CAL. Ce sont les avantages nécessaires dont ont besoin les ordinateurs ayant des performances réduites et des capacités de stockage faible.

Le CANopen est, par conséquent, une couche application standardisée par les spécifications du CIA (CAN In Automation) : DS-201…DS-207.

Le gestionnaire du réseau permet une initialisation simplifiée du réseau. Le réseau peut être étendu avec tous les composants que l'utilisateur désire.

Le bus CAN est un bus multi-maître. Contrairement aux autre bus de terrain, ce sont les messages qui sont identifiés et non les modules connectés. Les éléments du réseau sont autorisés à envoyer leurs messages à chaque fois que le bus est libre. Les conflits sur le bus sont résolus par un niveau de priorité donné aux messages. Le bus CAN émet des messages qui sont divisés en 2032 niveaux de priorités. Tous les éléments du réseau ont les mêmes droits et donc cette communication n'est seulement possible que sans bus maître.

Chaque élément décide lui-même lorsqu'il veut envoyer des données. Il est cependant possible de faire envoyer des données par un autre élément. Cette demande est effectuée par la trame distante.

Les spécifications du CANopen (DS-201…DS-207) définissent les caractéristiques techniques et fonctionnelles que nécessite un appareil individuel pour être associé sur le réseau. Le bus CANopen fait une distinction entre les appareils serveurs et les appareils clients.

- **La communication CANopen :**

Le profil de la communication du CANopen permet de spécifier les informations pour l’échange de données en temps réel et des paramètres. Le CANopen utilise des services optimisés suivant les différentes sortes de données.

↳ PDO (Process Data Object)

⇨Echange de données en temps réel

⇨Identifiant à haute priorité

⇨ Transmission synchrone ou asynchrone

⇨ Maximum de 8 octets (un message)

⇨ Format prédéfini

↳ SDO (Service Data Object)

⇨ Accède au dictionnaire des objets d’un appareil

⇨ Identifiant à basse priorité

⇨ Transmission asynchrone

⇨ Données distribuées dans plusieurs télégrammes

⇨ Données adressées par un index

Les caractéristiques diffusées par le CAN sont reçues et évaluées par tous les appareils connectés. Chaque service d’un appareil CAN est paramétré par un COBID (Communication OBject Identifier). Le COBID est un identifiant qui caractérise le message. C’est ce paramètre qui permet d’indiquer à un appareil si le message doit être traité. Pour chaque service (PDO ou SDO), il est nécessaire de spécifier un COBID à l’émission (envoi d’un message) et un COBID à la réception (récupération de message). Pour le premier SDO serveur, le COBID est fixe est ne peut pas être modifié à distance. De plus, il est calculé à partir du NODE-ID. Le NODE-ID est le paramètre qui caractérise l’appareil et qui permet d’accéder de façon unique à l’appareil.

PDO (Process Data Object)

C’est un échange de donnée arbitré entre deux modules. Les PDO peuvent transférer alternativement des synchronisations ou des événements contrôlés pour réaliser la demande d’envoi des messages. Avec le mode d’événements contrôlés, la charge du bus peut être réduite au minimum. Un appareil peut donc réaliser une communication à haute performance avec un faible taux de transfert.

L’échange de données avec le PDO utilise les avantages du CAN :

↳ L’envoi de message peut être déclenché par un événement asynchrone. (Événements contrôlés)

↳ L'envoi de message peut être déclenché sur la réception d'un événement de synchronisation.

↳ Récupération par une trame à distance.

SDO (Service Data Object)

C'est un échange de données point à point. Un appareil vient faire une demande d'accès dans la liste d'objets d'un SDO. Le SDO renvoie une information correspondant au type de requête fait par le demandeur. Chaque SDO peut être client et/ou serveur. Un SDO serveur ne peut pas faire de demande envers un autre SDO par contre lui peut répondre à toute demande d'un SDO client. Contrairement aux PDO, les SDO doivent suivre un protocole de communication particulier. La trame envoyée est composée de 8 octets :

↳ Domain Protocol (Octet 0) : il définit la commande (Upload, Download,….)

↳ Index sur 16 bits (Octet 1 et 2) : il définit l'adresse du dictionnaire des objets

↳ Sub-index sur 8 bits (Octet 3) : il définit l'élément de l'objet sélectionné dans le dictionnaire

↳ Paramètre (Octet 4 à 7) : Il définit la valeur du paramètre lu ou écrit.

Le gestionnaire de réseau comporte un mode simplifié de démarrage du réseau. La configuration du réseau n'est pas nécessaire dans tous les cas. La configuration par défaut des paramètres est donc parfois suffisante. Si l'utilisateur désire optimiser le réseau CANOpen ou augmenter ses fonctionnalités, il peut alors modifier lui-même ces paramètres. Dans les réseaux CANOpen, tous les appareils ont les mêmes droits et l'échange des données est directement régulé entre chaque appareil participant.

Le profil d'un appareil définit les paramètres nécessaires pour une communication. Le contenu de ce profil est spécifié par le constructeur. Les appareils ayant le même profil sont directement interchangeables. La plupart des paramètres sont décrits par le constructeur. Le profil possède aussi des emplacements vides qui correspondent aux futures extensions de fonctionnalités des constructeurs.

Dans la plupart des bus maître/esclave, l'efficacité du maître détermine le comportement de tout le réseau. De plus, les esclaves ne peuvent pas directement communiquer entre eux. Toutes ces caractéristiques augmentent donc, le nombre d'erreurs de transmission. CANOpen élimine tous ces désavantages. Le comportement temporel peut être spécifié individuellement pour chaque tâche respective des appareils participants. Ainsi, le système entier de communication n'a pas besoin de plus d'efficacité si seulement certains appareils participants nécessitent plus de performance. De plus, une tâche automatique peut être séparée pour chacun des appareils participants. Ainsi, les performances disponibles du contrôleur du réseau peuvent être utilisées de manière optimales et peuvent être augmentées à tout instant par adjonction de nouveaux appareils participants.

10-2-2- Dictionnaire

Le variateur gère uniquement le mode SDO pour accéder en lecture / écriture aux paramètres et aux variables.

Le dictionnaire contient les différents paramètres et variables du variateur.

Voir contenu du fichier : \DPL\DATA\ Modbus et CANopen.xls (A ouvrir de préférence sous Excel)

Parameter list of MD series								
OS Version : 1.0.3.0								
					Lecture CANOpen		Ecriture CANOpen	
Adress	Modbus Adr	Name	Decription	Size	Read Index	Read SubIndex	Write Index	Write SubIndex
600	400601	_RESTART_DRIVE		1	24673	0	8192	0
601	400602	_PARAM_MODE	Mode	1	24673	0	24672	0
602	400603	_PARAM_DRIVE_MODEL	Modèle	1	25872	1	25872	1
603	400604	_PARAM_DRIVE_NODE	Node ID	1	25872	5	25872	5
604	400605	_PARAM_DRIVE_I_NOM	Courant nominal	2	25872	99	25872	99
606	400607	_PARAM_DRIVE_I_MAX	Courant max	2	25872	100	25872	100
608	400609	_PARAM_I_NOM	Courant nominal	2	24693	0	24693	0
610	400611	_PARAM_I_MAX	Courant max	1	24691	0	24691	0
611	400612	_PARAM_TORQUE_NOM	Couple nominal	2	24694	0	24694	0
613	400614	_PARAM_MAX_MOTOR_S	Vitesse maxi	1	24704	0	24704	0
614	400615	_PARAM_MOTOR_PAIR_I	Nombre de paire	1	24653	0	24653	0
615	400616	_PARAM_SENSOR_TYPE	Capteur de temp	1	25616	32	25616	32
616	400617	_PARAM_RESOLVER_PA	Nombre de paire	1	25616	17	0	0
617	400618	_PARAM_DEPHASAGE	Déphasage / mo	1	25616	16	25616	16
618	400619	_PARAM_EXCITATION	Calage excitatio	1	25616	18	25616	18
619	400620	_PARAM_CALAGE	Calage automati	1	25616	19	25616	19
620	400621	_PARAM_GAIN	Gain excitation	1	25616	20	25616	20
621	400622	_PARAM_FILTRE_RESOL	Retour résolveur	1	25872	16	25872	17
622	400623	_PARAM_FILTRE_RESOL	Consigne ana filt	1	25872	17	25872	17
623	400624	_PARAM_EMULE_CODEU	Emulation codeu	1	25872	32	25872	32
624	400625	_PARAM_CODEUR_RES(	Résolution code	2	25872	33	25872	33
626	400627	_PARAM_CODEUR_NUMI	Numérateur	1	25872	34	25872	34
627	400628	_PARAM_CODEUR_DIVIS	Diviseur	1	25872	35	25872	35
628	400629	_PARAM_INPUT_INVERT	Inversion des ent	1	25872	48	25872	48
629	400630	_PARAM_INPUT_FILTER_	Activation filtre	1	25872	49	25872	49
630	400631	_PARAM_INPUT_FILTER_	Période filtre	1	25872	50	25872	50

- Variable flag :

  On échange 16 bits à la fois sous forme d'une variable entière.

  Ex : Index 12288, Sous index 0 correspond à VF0 à VF15

- Variable octet :

  On échange 2 octets à la fois sous forme d'une variable entière.

  Ex : Index 12544, Sous index 0 correspond à VB0 à VB1

- Variable entier :

  Le type échangé est le même.

- Variable entier long :

  Le type échangé est le même.

- Variable réelle :

  Les valeurs envoyées devront être cohérentes avec l'unité et le nombre de décimales (précision) paramétrées dans le logiciel à partir du menu Options / Langage DPL / Compilateur.

  Exemple : Précision paramétrée de 0,01

  Unité : mm

  On souhaite charger 100.5mm dans la variable VR0

  Index 13312, Sous index 0, Valeur 10050

  WriteParam (13312,0) = 10050

  VR1 = ReadParam (13312,0) 'est équivalent à VR1 = VR0

  Voir : WRITEPARAM, READPARAM

10-3- MODbus :

10-3-1- Définition :

Le protocole MODBUS est un protocole maître/esclave utilisé principalement dans le milieu industriel. Il permet à des équipements de supervision (Human Machine Interface, Supervisory Control And Data Acquisition), de communiquer avec un ou plusieurs équipements industriels (Programmable Logic Controllers, automates,sondes, etc..).

Ce protocole fonctionne sous forme de requête. Ces messages transitent sur un support physique qui peut être une liaison asynchrone RS232, RS422 ou RS485.

Pour distinguer un équipement esclave d'un autre, on attribue un numéro d'identification (Unit ID) à chaque équipement. Grâce à ce numéro et dans le cas d'une liaison un à plusieurs (cas du RS485) seul l'équipement esclave concerné répondra à une requête d'un équipement maître.

Le variateur gère le protocole MODBUS RTU Esclave.

Le format de la liaison est 8 bits de données, 1bit de stop et pas de parité.

La vitesse de transmission peut aller jusqu'à 57600 bauds

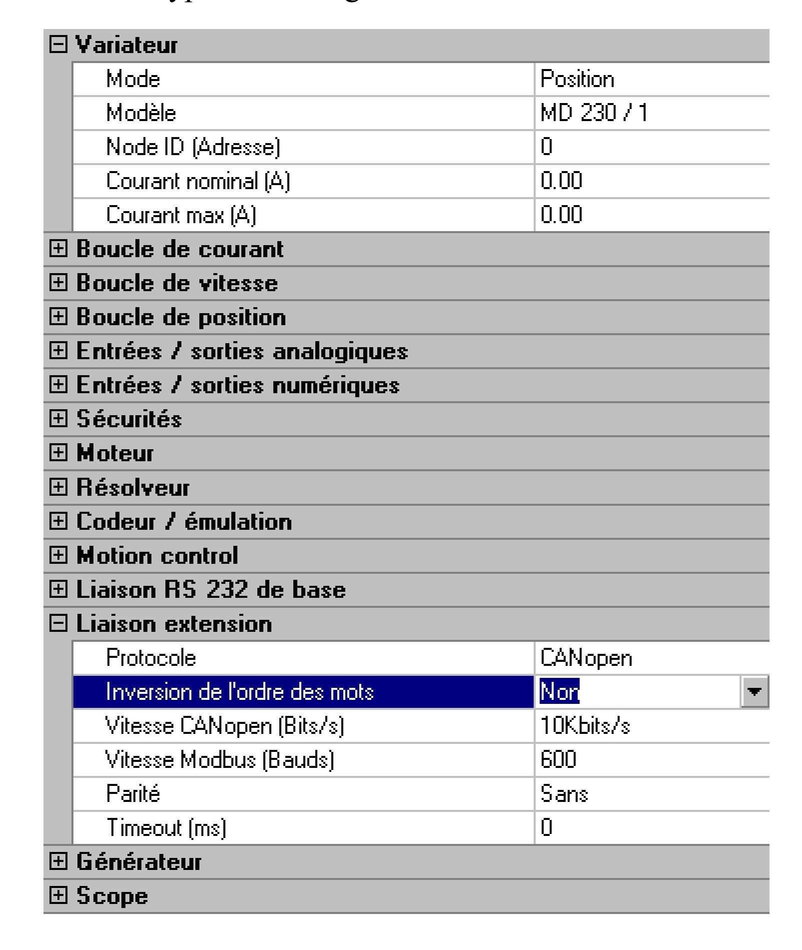
Les fonctions de lecture de mots (fonction n°3 ou 4) et écriture des mots (fonction n°16) sont reconnues par le variateur.

10-3-2- Variables codées sur 2 mots

Les paramètres du variateur ainsi que les variables de type entier long et réel sont codés sur 2 mots (32bits). Comme l'indique la norme Modbus, un double mot est de la forme suivant :

Adresse :	Mot :
n	Poids fort
n+1	Poids faible

Le paramètre « Inversion de l'ordre des mots » accessible à partir de la liste des paramètres dans le groupe Liaison extension permet d'inverser le codage du double mot sur les variables de type entier long et réel.



L'inversion n'a aucun effet sur les paramètres du variateur qui sont toujours codés suivant la norme poids fort, poids faible.

Liaison :	Type liaision system	Paramètre inversion	Inversion Codage VR et VL	Inversion Codage paramètres
RS232 de base (Connecteur X1)	Activé	X *	Non	Non
RS232 de base (Connecteur X1)	Non activé	Non	Non	Non
RS232 de base (Connecteur X1)	Non activé	Oui	Oui	Non
Extension Modbus (Connecteur X4)	X *	Non	Non	Non
Extension Modbus (Connecteur X4)	X *	Oui	Oui	Non

* X : état indifférent

Si Inversion codage = NON ⇒ Adresse n : poids fort

Adresse n+1 : poids faible

Si Inversion codage = OUI ⇒ Adresse n : poids faible

Adresse n+1 : poids faible

10-3-3- Dictionnaire

Le dictionnaire contient les différents paramètres et variables du variateur.

Voir contenu du fichier ..\DPL\DATA\ Modbus and CANopen.xls (À ouvrir de préférence sous Excel)

- Paramètres accessibles entre les adresses 600 et 900
- Variables type flag sont accessibles entres les adresses 57344 et 57359
- Variables type octet sont accessibles entres les adresses 57360 et 57487
- Variables type entier sont accessibles entres les adresses 57388 et 57743
- Variables type entier long sont accessibles entres les adresses 57744 et 58254
- Variables type réel sont accessibles entres les adresses 58256 et 58767